BY M. R. SHURA-BURA, ET AL.

29 JUNE 1979 FOUO 1 OF 2

JPRS L/8555

29 June 1979

[redacted]

USSR

[redacted]

[redacted]

DOS YES UNIFIED OPERATING SYSTEM GENERAL PRINCIPLES

By

M. R. Shura-Bura, et al.

U. S. JOINT PUBLICATIONS RESEARCH SERVICE

[redacted]

CONTENTS (Continued)

DOS YES UNIFIED DISC OPERATING SYSTEM. GENERAL PRINCIPLES

Moscow OPERATSIONNAYA SISTEM DOS YES (OBSHCHIYE POLOZHENIYA) in Russian
1975 pp 1-120 signed to press 4 Jul 75

[Book by M. R. Shura-Bura, E. V. Kovalevich, M. S. Margolin, M. G. Skoromnik, L. T. Chuprigina, Statistika Publishing House, 80,000 copies]

[Text] Abstract

This book is an introduction to the DOS YeS unified disc operating system which is used in the latest models of the unified computer system created by the joint efforts of the participating countries of the CEMA. The authors familiarize the reader with the composition, the structure and the characteristic features of the system. A highly detailed description is presented of the functional capabilities of the components of the unified disc operating system and their interrelation.

The book is of interest for users of the unified disc operating system; it can be used both for independent study and as a text for the programming instructors.

FOREWORD

The software for a computer is the set of programming means, rules, procedures permitting use of the computer for the solutions of various problems. Only in the presence of a sufficiently developed software will the hardware making up the modern computer be an extraordinary tool for the solution of all possible information processing problems.

Modern software includes the set of programs facilitating and automating a significant part of the operations connected with using computers. Among these programs the primary role is played by the control programs which organize the flow of the problems in the machine by efficient distribution of the reserves having error detection means for the operation of the equipment and errors in the programs. An important property of the modern control programs is its capacity for making intelligent, independent decisions on occurrence of various situations, including the appearance of errors, during the operation of the computer, combined with the presence of means of communicating with the operator, means permitting the operator to input to the control tactics and in a number of cases to make the final decisions.

An important part of the software is the programs which simplify and automate the process of compiling and checking out the programs. These are primarily compilers which offer the possibility of presentation of the initial program in the corresponding algorithmic language and the library of programs and logarithms permitting formulation of the initial assignment in terms of quite large operations. The significance of these libraries is increased for the presence in them of software which realizes the compilation of the programs from the previously prepared parts by assembly and editing. In recent years a larger and larger role has been played in software by the packages of applied programs which are problem-oriented libraries of programs, supplemented by access and control means. This package permits us to combine simplicity of formulation of the assignment to compile the program for the solution of any specific problem from the corresponding class of problems with high efficiency of the program obtained.

For the first generation computers, in particular, for the second generation, the storage of separate sets of programs and making them available to the

user were characteristic. The low equipment reliability and small memory size, especially the size of the direct-access memory, established very rigid frames for the development of software and especially for the creation of complex interdependent systems which offer an all-around solution to the problem of supporting the operation of the computer. The same causes also prevented the application of the computer for complete automation of the information processing when working with large data files. The computer was used only for fragments or individual stages of these processing processes.

The technological advancement and significant improvement in equipment reliability, the growth of the ready-access memory size and the appearance of fast, capacious direct-access memories -- magnetic discs -- have created a situation in which it turned out to be possible and necessary to have an all-around solution to the problems of organizing the operation of the computer excluding faults on transferring from one phase of solving the problem to another or when changing problems, leading to minimum manual manipulation and insuring a high equipment utilization factor. The nature of the software accumulated at that time created serious difficulties for the implementation of this all-around approach. The establishment of the languages and interaction among the various parts of the software turned out to be complex problems, for each part was created independently without considering the peculiarities or even the existence of other parts. Even in the case of independent use of different parts of the stored software, the user and the computer personnel experienced serious difficulties. The same or analogous problems were solved in different chast differently, and it was necessary for the user to remember the entire variety of methods of organization and forms of access, and it was necessary for the personnel to deal with many carriers from which each required its own method of use and accompaniment.

The efforts to convert the separate parts of the software into a unified sytem by creating all-possible linking programs and partial reworking of some of the existing ones, could not seriously advance the solution of the problem. It became clear that the basis for the computer software must be a common design for a software system providing not only for the functions, composition and organization of the future system, but also the paths of future development of it. The appearance of these designs and the work on implementation of them can be considered the birth of systems programming. The sets of programs created by these designs have come to be called operation systems.

The first efforts to create a system functionally combining the various parts of the software were made at the beginning of the 1960's. One of the first was the system which was developed for the ATLAS computers. A basic characteristic of this system consists in the fact that all of the programs -- compilers, applied programs, interrupt process programs -- are logically subroutines of a supervisor (controlling program), the operation of which is initiated by the supervisor by transferring control to it, and each program, on completion of operation, returns control to the supervisor.

The packaged processing of the assignments with planning of the mixture of problems coming into the system, multiprogram, the input-output control system with logical addressing of the peripheral devices and dynamic memory distribution were implemented in the system.

Among the first systems let us also note the IBSIS system for the IBM-7090/7094 computers. This system is built around the primary dispatcher who controls the system by means of his branch subroutines and the subordinate dispatcher. The system includes translators from Cobol, Fortran, Macroassembler, the loading and sorting programs and the editor. The programming standards make it possible to have a common library of programs. The adaptive input-output control system permits the working programs to be equipped with the necessary data control subroutines. The system is open to be supplemented by translators and other processing programs.

These systems were direct predecessors of the OS/360 system which was announced in 1964 and up to now has been one of the most powerful and developed operating systems.

The development of the operating systems and the functions performed by them have had a noticeable influence on the architecture and operating logic of modern computers. Now it is in practice impossible to solve even the simplest problems on a computer without the operating system. The operating system is a necessary attribute of the computer.

In the last 10 years, the concept of the operating system has been transformed somewhat. Occurring as a synonym of the combination of software into a system, the term "operating system" has naturally come to be used to designate only part of the system, including and providing the most used all-purpose media.

For example, the packages of applied programs operating under the control of and with the help of a given operating system usually are not considered components of the operating system although they are an extension of it, essentially supplementing the set of services made available to the user by the system.

Frequently in the literature the term "operating system" is used in a narrower sense, including only the designation of the system of programs providing for execution of finished programs and isolating the means of automating the preparation of programs in an independent concept -- the programming system.

Here, however, we shall adhere to the broader interpretation of the "operating system," including in it the program compiling means.

In almost any sphere of human activity when using any new untested means, the matter begins with very modest scales and only as progress is made do the requests for this means and skills of application grow. For the

process of introducing computer engineering in one area or another or even simply for the development of a computer center designed to service the required computer means, an increase in the requirements on the capacity of the computer means and on the variety of equipment is characteristic on the one hand, and on the other hand, the necessity for retaining the stored programs and their use under the new equipment conditions. One of the methods of using these requirements is the creation of a family of program-compatible machines of different speed and a broad set of peripheral devices with standardized method of connecting them to the various computers of the family. The user can begin operations, having at his disposal a minimum of equipment required by him, building it up as the demands increase or proceeding to the use of a more powerful model within the family. The software of such a family of computers can satisfy the additional requirements of adaptation to operation with various sets of specific devices. In order to achieve efficiency and insure the greatest capability of functioning on devices with comparatively modest memory sizes, this adaptation basically takes place statically, that is, by preliminary adjustment. The operating system of any specific device appears as a result of a special process called generation which is carried out with respect to the programs available in the software.

By the operating system of the family of computers in this case we mean both the set of all specific operating systems created during generation and the set of still untuned moduli of the system and the generation programs, the presence of which offers the possibility of obtaining any specific operating system. This duality of the term constantly must be kept in mind in connection with the DOS YeS unified disc operating system described below.

In spite of the attractiveness and the prospectiveness of this universal approach in which the operating systems for the different machines in different combinations are versions of the unified system, its effective implementation in full volume appears to be highly problematic at least for the family of computers in which there are low-efficiency models and provisions are made for sets below some minimum. The problems of distribution of resources, planning of the mixture of problems, organization of parallel processes inside one problem, completeness of diagnostics, stability with respect to errors and erroneous operations, the exclusion of manual manipulations and so on which are decisive for the efficiency of the operating systems of powerful computer complexes are for various reasons pushed into second place for the computer with modest resources. For such computers the primary importance is attached to the problems pertaining to the organization of input-output, the structure of the programs and program library, convenience of manual manipulations, and so on and, of course, the minimization of the resources taken out by the operating system, the speed and simplicity of its operating algorithms do not play the least important role. At the same time an important role is played by such modest installations among the computers which are produced and in operation at the present time.

One of the most widespread operating systems oriented toward such installations is the DOS/360 created for the efficient utilization of the most recent models of the 360 series computers built by IBM. With respect to the basic structural principles, the DOS/360 is similar to the more powerful OS/360, but it takes into account the limited possibilities of the equipment of the smaller machines (the small basic memory, low processor speed, and so on). However, the DOS/360 retains its fitness and can be used on all models of the 360 series.

The operating systems of the DOS/360 and the OS/360 have had great effect on the development of the operating systems, serving as a model for many both as a result of the widespread use of the series 360 machines and on the basis of their undisputed advantages.

The DOS YeS operating system used on the models of the YeS-1020, YeS-1030 and YeS-1040 of the unified system of computers developed by the joint efforts of the member countries of the CEMA, is functionally similar to the DOS/360 system. This similarity permits exchange of the applied programs and other information among the users of the series 360 and YeS computers.

This book contains a description of their purpose, the possibilities and structure of the DOS YeS unified disc operating system.

CHAPTER 1. BASIC CONCEPTS

The unified computer system (YeS EVM) is a number of program-compatible models of different output capacity, each of which includes a processor, a basic memory, input-output channels and a set of various peripheral devices. The coupling of the processor and the basic memory to the devices is via channels. The set of the specific installation of the unified system of computers can vary widely as a result of the basic memory size, the channels and the set of peripheral devices.

The DOS YeS unified disc operating system is a set of means and programs which permit us to obtain a version of the DOS YeS operating system for each admissible combination of equipment which provides for operation of the unified computer system installation in a given complex. The operating conditions provided for by the operating system and the set of properties and functions included in it can vary as the user desires in accordance with his needs.

The DOS YeS system makes it possible for the user to prepare his program and solve his problem with it to a significant degree independently of what specific installation of the integrated system of computers and with what version of the DOS YeS system these operations are performed. For the majority of users the DOS YeS system is a unified operating system permitting automation of the entire process of setting up the problem on the YeS EVM unified system computer beginning with planning of the structure of the program and the encoding and ending with correct execution of the program.

The DOS YeS system removes the necessity for detailed description of all the operations which must be performed by the computer for the solution of the problem from the programmer, and at the same time decisively simplify the programming. The set of program components of the DOS YeS is made up of the modules of the unified system combined by common principles of construction and interaction for all having matched inputs and outputs. The coordination center of the system is the monitor, under the observation of which all the programs are executed, both the operating system itself and the programs of the user. The list of most significant properties of the DOS YeS operating system is matched with more precise determination of some of the general concepts and terms which will be used hereafter.

The Multiprogram Processing

The problems can be solved on the computer in the single-program or multiprogram regime. In the single-program regime the computer solves only one problem and proceeds with the solution of the next only after completion of the preceding one. The solved problem can be granted all of the available resources without which the operating system can get along by itself. In the multiprogram regime there can be several problems in the stage of execution on the computer, and the operating system must distribute the resources during the corresponding programs.

Under the control of the DOS YeS operating system, the programs can be performed both in the single-program and in the multiprogram regime. In the multiprogram regime the DOS YeS insures the possibility of simultaneous execution of up to three programs which do not depend on each other.

Assignment

In order that the computer perform the operation, the operating system must obtain the corresponding assignment. Modern computers are rarely used for single operation. As a rule, the computers are used to process the flow (or flows) of assignments. These can be assignments with the previously planned performance sequence or the assignments arriving asynchronously from the different objects. The assignments themselves can be of different urgency and importance.

The operating system must organize the acceptance of the assignments, monitoring of them, the creation of the sequences for execution, the preparation of the requested programs for execution, starting of them, automatic transfer to the next assignment. These functions are performed by the programs of the operating system having the common name -- Assignment Control. The Assignment Control programs provide in the system for the acceptance of a request to perform an operation and transfer to the execution of a new operation as soon as it becomes possible. The basic purpose of these programs is the organization of the automatic transition from one assignment to another, which requires automatic adjustment of the system on making the transition to the new assignment.

For normal operation of the DOS YeS system, the assignments must be grouped in advance into a package which forms the input flow of assignments. The operator initiates the reception of the given flow, reporting from which device it must be read. The assignments from the input flow are received and interpreted by the Assignment Control program and are executed automatically in succession, one after the other, without intervention of the operator or programmer. This operation of the system where the assignments are received and are executed successively is called package processing.

In the DOS YeS system, the assignment is described using the controlling operator. The assignment usually consists in several controlling operators,

determining, for example, the names of the assignment, the beginning and end of the assignment, the name of the program which must be executed. The assignments combined into one package are not related to each other, and the execution of one assignment does not depend on the other assignment.

The assignment can be divided into steps. The steps of one assignment are executed in series. The execution of each next step of the assignment, as a rule, depends on the successful execution of one or several preceding steps. For example, the assignment for an observation including translation, editing and execution of the program can be made up of three steps: the performance of the required translator, the execution of the editor program, the execution of the program directly (translated and edited).

Division

The problem of the distribution of resources between the simultaneously executed programs has been simplified in the DOS YeS by preliminary breakdown of the basic memory into divisions and attachment of certain input-output devices to each division. The flow of assignments begins with one of the divisions, and all of the assignments (and steps of the assignments) from the given flow are executed in series, using the corresponding division together with the devices attached to it.

The type of multiprogram processing adopted in the system is called multiprogram processing with a fixed number of divisions. One part of the basic memory is always occupied by the control program. This part is called the control program region. The rest of the basic memory is called the problem program region. The programs written by the user and such programs of the operating system as the translators, sorting and copy programs are executed in this region.

The DOS YeS operating system permits representation of the problem program region in the form of one, two or three divisions. Each division is designed for the execution of one program. If the problem program region is made up of one division, only one program at a time can be executed on the computer. If the problem program region is broken down into two or three divisions, then two or three programs can be executed simultaneously.

File

In the DOS YeS operating system, the data which are on the external carrier and are read from the input device during processing or are recorded on the output device constitute files. A file can be located in any input-output structure of the unified system of computers. Depending on the type of device and the program requirements a file can have various sizes and different structure. The file concept is analogous to the concept of the "data file." However in the DOS YeS the file is a concept which defines any information which is placed on an external carrier, whether it is programs, assignments or data. For example, the input flow of assignments is a file; the initial program for the translator is a file; the data to be sorted also constitute a file.

The external carriers on which the files appear are called volumes. For example, a volume is a magnetic tape reel or package of discs. One volume can contain several files, forming a multifile volume, and one file can be placed on several volumes, forming a multivolume file.

The characteristics of the data file which permit the system to recognize this file, such as the name of the file, its limits, format, and so on, are contained in special modules called the file markers. The file markers, as a rule, are placed on the same carriers as the files themselves.

Logical Devices

The YeS EVM unified system of computers permit connection of a large number of different input-output devices. On the different computers the composition of the peripheral devices is different. The physical addresses of the devices can be distinguished both on the computers having an identical set of devices and on the computers having a different set. In order to simplify the transfer of programs from one computer to another, from one configuration of the external device to another, it is necessary to remove the necessity for referencing specific physical addresses of the devices from the programmer.

The method of logical devices is used for this purpose in the DOS YeS system. When it is necessary to reference an external device in the program, the programmer indicates not the physical address of the device, but the symbolic name selected from the set of names given to the logical devices. Directly before execution of the program on the computer specific input-output devices must be placed in correspondence (assigned) to the logical devices used. The system takes in itself the conversion of the logical addresses to physical during the execution of the program. The number of logical devices can not coincide with the number of devices connected to the computer, for the same physical device can be designated, for example, as two logical devices.

There are defined rules for using symbolic names for logical devices. The system isolates two categories of logical devices: the logical devices of the programmer and system logical devices. There are differences in the use of the indicated categories of logical devices. The logical devices of the program are used in the user programs with respect to an examination of the programmer, and the system permits each of them to call any of the input-output devices. The system logical devices are used by components of the DOS YeS operating system itself for its own purposes. Specific names are fixed for these devices, and each system device can have one or several specific input-output devices of the admissible types assigned to it. The basic system logical devices are the following:

The system residence (SYSRES) -- the memory on discs on which the DOS YeS system itself is located;

The system read driver (SYSRDR) -- the device used for input of the controlling operators of the assignments;

The system input (SYSIPT) -- the device used for data input;

The system punch (SYSPCH) -- the device used for data output in card format;

System printer (SYSLST) -- the device used for printing the data out;

The system register (SYSLOG) -- the device for communication with the operator.

There are also other logical system devices.

Steps in Preparing the Programs

An important role in the operating system is played by the means for automation of programming which includes the programming languages and translators, the programs providing for the combining and editing of various programs and debugging means. During preparation of the program in the DOS YeS, three basic steps are isolated: compiling of the program (writing the program) in the initial programming language, translation and editing.

The DOS YeS operating system allows the programmer the programming languages oriented toward different classes of problems: Fortran -- for scientific and technical problems, RPG -- for accounting problems, Cobol -- for problems in economics, PL/1 -- the universal programming language, the Assembler -- the machine-oriented language. The Assembler permits the use of all the capability of the machine and at the same time relieves the programmer of such fatiguing and tedious tasks as remembering the machine codes of the instructions, calculating the true addresses and memory, and so on.

The program written in any of the program languages is called an initial module or initial program. The initial program is converted by the corresponding translator to the target module. The target module is the program module in intermediate format common to all the system translators. The target modules are still not machine programs. They are not designed for direct execution, for in addition to the program text, they contain additional information permitting organization of communications among the modules, placement of the program in a specific location in the base memory, and so on. The target modules must go through another processing step -- the editing step. In this step the modules are processed by the editor program. As a result, a program is obtained which is ready for execution. It is a set of program phases (or one phase). The phase also called an absolute module, can be loaded in the basic memory for execution. In the DOS YeS the phase is loaded in the basic memory for execution always only as a whole.

In the editing step the program phases can be collected from individually prepared modules, and the finished modules available in the libraries can be included in them. The combination of the target modules into phases takes place directly when the module is obtained from each programming language.

The preparation of the program with respect to steps permits the programmer to make flexible use of the system means. The programmer can break down his program into parts, and he can program them individually (frequently this is done by different programmers). For each part a specific programming language can be selected; the parts can be translated independently and checked out autonomously. This breakdown of the program into modules permits if necessary the introduction of changes only into individual parts of the program without touching the rest. The communications between the individually prepared parts of the programs (modules) are defined symbolically, and specific values are obtained only in the editing step.

Structure of the Program

In the DOS YeS, a program can have one of the following structures: simple with overlay without a base phase, with overlay with a base phase:

The simple structure predetermines that the program is made up of one phase, that is, the program will be called into the basic memory as a whole for execution.

The structure with overlay without a base phase resembles the program execution conditions under which each executed phase calls the next phase into the basic memory in its place. Thus, the called phase completely or partially occupies the space which was previously occupied by the calling phase. A section equal to the length of the largest of the phases is occupied in memory.

The overlay structure with a base phase presupposes that there is a phase which is constantly located in the basic memory during execution of the program -- the base phase. The remaining phases are called into the basic memory by the base phase alternately in the same location.

The structures with overlay permit the solution of the problems requiring large basic memory size.

For segmentation of planning of the program structure, it is necessary to know the logic of the program in the links between its parts well. This information cannot always be transferred to the operating system, therefore the programmer himself must deal with the problems of segmentation and planning. For these purposes the system equips him with the corresponding apparatus for external communications instead of directives for assembly and calling of the phases.

System Generation

An important property of the DOS YeS operating system is its adaptability to the conditions of different applications, the methods of operation and configuration of the technical means. This adaptability is insured by an organic combination of various means, the most important of which must be considered to be the modularity of the system, the presence of macrogeneration, independence of the programs with respect to specific addresses, and in individual cases, the characteristics of the external devices, simplicity of modification, and supplementing.

The process of obtaining a specific system from the general DOS YeS is called system generation. Generation includes the following:

Adjustment of the control program (primarily the Supervisor) to insure the specific composition of the equipment (the peripheral devices, the model number, the volume of the basic memory, and so on) and also introduction into the generator system as the user requires such properties as the number of divisions for multiprogram processing, memory protection, the statistics on magnetic tape failures, and so on;

Planning the dimensions and composition of the libraries, the operating files of the system programs, standard designations of the devices, and so on.

The DOS YeS obtained by the user for his specific purposes during operation of the system is permanently placed on discs. Therefore the computer on which the DOS YeS is used must contain at least one disc memory.

Composition of the DOS YeS System

The DOS YeS system is a broad program complex including programs for different purposes. These are programs that organize and control the computing process, the translators from the programming languages, and the programs that service the system itself. Considering the purposeful direction of the programs, the developed traditions for classifying them and also the convenience of discussing the material, the DOS YeS programs can be represented in the form of three basic groups of programs: the monitor, the control of the data and the processing program (Fig 1).

The monitor insures automatic control of the computer and is made up of several independent programs;

The initial load program which does the initial preparation of the system for operation;

The supervisor, under the supervision of which all the programs are executed, beginning with the time of input of the assignment to execution of the program and obtaining of the results;

The Assignment Control program which prepares the system for execution of each individual assignment.

The data control deals with organizing the data and processing of it. These problems are solved on the level of each programming language individually. In the Cobol, Fortran, RPG, and PL/1 programming languages, the data control means are embodied in the operators of the language itself. For the programmer who writes his own programs in the DOS YeS Assembler language, the data control means are represented in the form of a set of programs forming the input-output control system.

The processing programs include transmitters from the programming languages Assembler, Fortran, RPG, PL/1 and Cohol, the set of service programs and also the user programs.

The servicing programs include the following:

The Editor which shapes the program ready for execution on the computer from the target modules;

The Librarian which provides for updating and servicing the DOS YeS libraries;

The Debugger, which provides the programmer with the means of debugging the programs;

The copy programs providing for encoding, shifting, printing and punching the files and other auxiliary functions;

The sorting programs;

The program for checking the peripheral devices, which checks the peripheral devices for busy simultaneously with the execution of other programs.

All of the program components of the DOS YeS and also the programs created by the user are stored in the system libraries. There are libraries of three types:

The library of initial modules containing the program modules written in the initial programming languages;

The library of target modules containing the target modules which are the results of the translation of the initial modules;

The library of absolute modules containing the program phases ready for execution.

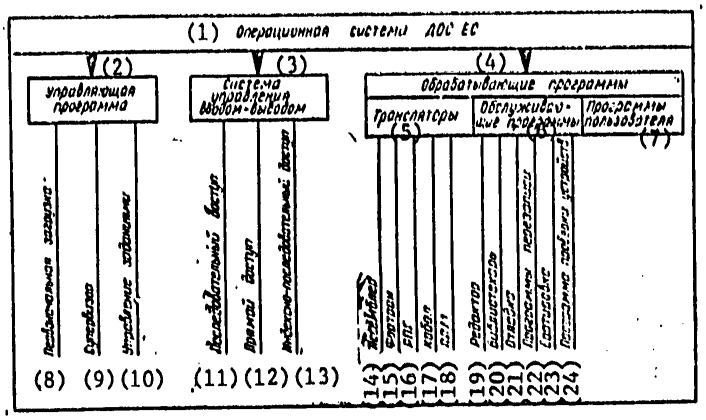

Figure 1. Composition of the DOS YeS

Key:

1. DOS YeS operating system
2. Controlling program
3. Input-output control system
4. Processing program
5. Translators
6. Service programs
7. User programs
8. Initial load
9. Supervisor
10. Assignment control
11. Series access
12. Direct access
13. Index-series access
14. Assembler
15. Fortran
16. RPG
17. Cobol
18. PL/1
19. Editor
20. Librarian
21. Debugger
22. Copy programs
23. Sorting
24. Program for checking the peripheral devices

CHAPTER 2. MONITOR

From the point of view of the user, the operating system is an extension of the hardware. The programs of the operating system which react to the interrupt signals are the closest to the hardware. These programs are part of the control program or monitor; their basic purpose is to supervise the execution of the processing programs. The monitor sees to the operation of the programs, it determines the order of their execution, it reacts to errors in the programs and hardware, it completes the operation of the programs, it sees that the programmers observe the limits of use of memory and time, and it monitors the use of the input-output units.

For the monitor it is insignificant which program is executed at a given time: the program entering into the DOS YeS composition or compiled by the user (the problem program). For the monitor they are all equal and equivalent to the problem program. By a problem program hereafter we mean a program written by a user for his problem. Only in cases where the situation is considered from the point of view of the monitor can we consider the problem program to be any processing program of the DOS YeS.

The monitor must constantly be in a fit condition as a necessary supplement to the hardware. Part of the basic and external memory are required for location of it. In the basic memory this is the monitor region; in the external memory, the system residence, that is, the packets of discs on which the monitor and the DOS YeS programs are permanently located. All the rest of memory (basic and external) is at the disposal of the user.

The monitor is made up of several independent programs which perform their own functions in the different steps of the computing process (see Fig 2). The composition of the monitor includes the following: the Supervisor, Assignment Control, Initial Load.

The Supervisor is the coordinating center for the monitor, which reacts to the interrupt signals, distributes the processor time among the programs being executed simultaneously, loads the required program for execution. The functions of the Supervisor are varied, and the volume of programs executing these functions is great. Therefore the Supervisor has a

structure with overlay with the base phase. The base phase of the Supervisor is called the nucleus. It is constantly present in the basic memory; one of its functions is to call the required parts of the monitor from the system residence into the basic memory. The monitor region has a special section where other phases of the monitor are called -- the transient zone. The phases called into the transient zone from the system residence are called transit or transits. The criterion for the division of the functions between the nucleus and the transits is the use frequency of the function and, especially, the admissible reaction time of the system for a request for the given function. The nucleus of the Supervisor analyzes the interrupt signals, the transit phase call, switching between programs in the multiprogram regime, and so on, that is, the functions which must be executed quickly and quite frequently. The processing of the errors from the peripheral devices, the organization of communications with the operator, creation of control points, and so on, that is, the functions, the performance of which is required quite rarely are transferred to the transits.

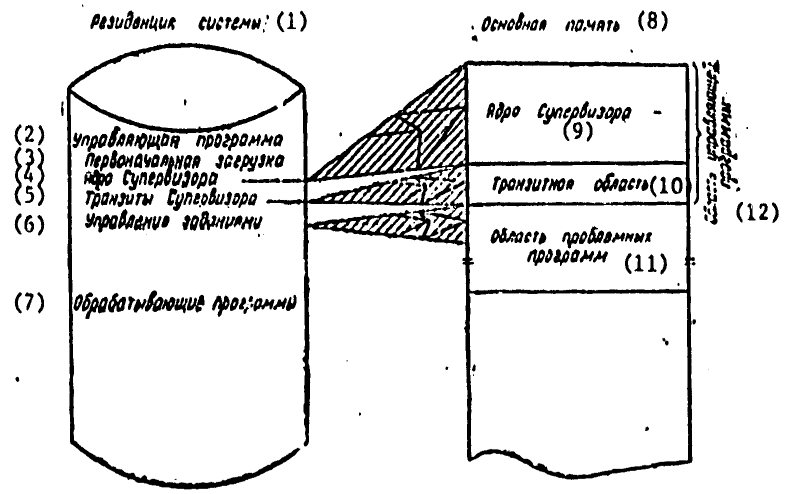

Figure 2. Monitor Composition

Key:

1. System residence
2. Monitor
3. Initial load
4. Supervisor nucleus
5. Supervisor transits
6. Assignment control
7. Processing programs
8. Basic memory
9. Supervisor nucleus
10. Transit zone
11. Problem program zone
12. Monitor zone

The Assignment Control executes the functions required to prepare the program for execution under the supervision of the Supervisor. This part of the monitor is called into the basic memory by the Supervisor from the system residence, and it functions with transition from execution of one problem program to another. The Assignment Control does not require additional room in the basic memory, for it is loaded into the memory zone in which the next assignment (or assignment step) will be performed. On completing the required preparation, the Assignment Control calls "on its own" the program which is required in the assignment, at the same time "destroying" itself in the basic memory.

The Initial Load program provides for the preparation of the DOS YeS system for operation. Its basic mission is to load the nucleus of the Supervisor from the system residence into the monitor zone.

Multiprogram Processing

The primary goal of multiprogram processing is to increase the carrying capacity of the system as a result of more efficient utilization of the computer resources. In the case of multiprogram processing the increase in output capacity is achieved as a result of the fact that simultaneous performance of the input-output operations and the processing operations is permitted. For example, at the time when one program performs the input-output operations, the other program can process the data in the basic memory.

In order for it to be impossible simultaneously to execute several programs, the system must solve the basic problems such as the distribution of the computer resources among the programs (basic memory and peripheral devices), the assignment and the maintenance of priorities. The problem of the distribution of resources is solved in the DOS YeS system as follows: the basic memory is divided into divisions and the peripheral devices are attached to each division. This distribution always exists for multiprogram processing, but on each machine this distribution can be specific to the machine; for a specific mixture of simultaneously executed programs it is also possible to establish a specific distribution.

Each program is executed in one of the basic memory divisions. Two or three programs which do not depend on each other can be executed simultaneously. The program ready for execution, as a rule, is adjusted to the division of the basic memory in which it must be executed. Therefore the separation of the basic memory into divisions must be considered in the program editing phase. When the program is to be executed in another division, it must be edited for this division.

The quantity and size of the divisions are determined during the process of generating the system or on execution of the initial loading procedure, but then they can be altered by the operator during operation of the system to satisfy the requirements of the individual problem programs.

The effectiveness of the multiprogram processing depends on how successfully the basic memory is distributed among the divisions. In order that the distribution be successful, it is necessary to plan it considering the requirements of many programs and not one specific assignment.

The divisions have names: background division, first division of the front plan and second division of the front plan. Correspondingly the programs executed in these divisions are called background and front plan. The background programs are executed in the background division, the front plan programs are executed in the front plan divisions. The system can process one background program and one or two front plan programs simultaneously (see Fig 3).

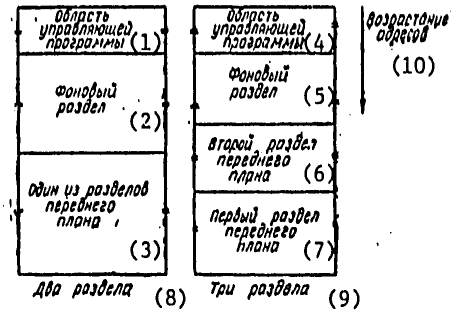

Figure 3. Distribution of the basic memory in two and three divisions during multiprogramming

Key:

1. Monitor zone
2. Background division
3. One of the divisions of the front plan
4. Monitor zone
5. Background division
6. Second division of the front plan
7. First division of the front plan
8. Two divisions
9. Three divisions
10. Increasing addresses

Each division must be considered not only as a defined zone of the basic memory, but also as a set of the basic memory zone and the external devices with which the program of the given division operates. All of the input-output devices connected to the computer are distributed among the divisions. The program executed, for example, in the background division can make use only of the input-output devices which are branched for the background division; analogously, the front plan program works with devices allocated for the corresponding division of the front plan.

Since in the initial program the programmer indicates the logical devices and not the specific physical input-output devices, it is important for him to know only which logical devices he can use in its program. Independently

of in which of these divisions (background or front plan) the program is executed, it can use both any logical devices of the programmer and the system logical devices. Some of the system logical devices cannot be used in the front plan programs. Therefore there are programs, including system programs which can be executed only in the background division. For example, the editor program and the DOS YeS translators are executed only in the background division.

The physical units for the used logical units are designated directly for fuller execution of the program. This designation is indicated in the assignment for execution of the program which does not change the program text. The designation of the specific physical device for the logical device of the given division means attachment of this physical device to the given division. The physical device belongs to the division until the assignment is cancelled. When the assignment is cancelled, this device can be allocated to another division.

One and the same physical device cannot be used simultaneously in different divisions. As an exception to this rule we have the disc memories and typewriter. In all of the divisions for the system residence the same disc memory is used, and as the device for coupling to the operator, the same panel typewriter. The system also permits the use of various sections of the same package of discs by the programs of the different divisions. However, if programs are executed simultaneously which use different sections of the same package of discs, the output capacity can drop as a result of an increase in search time for the required information.

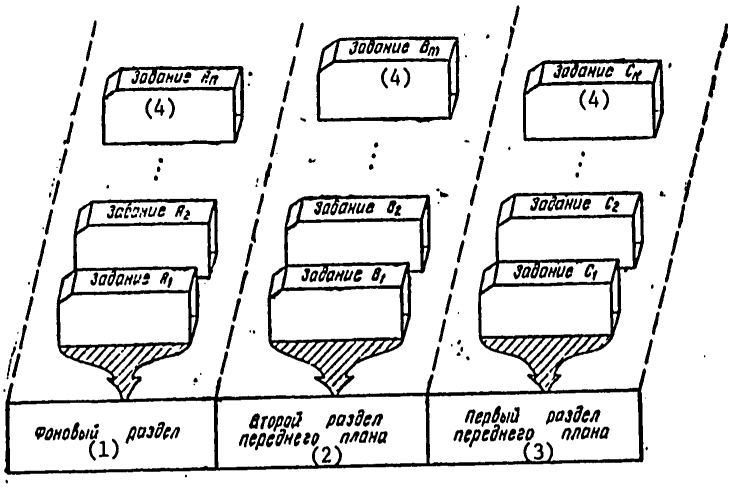

Figure 4. Input flows of assignments during multiprocessing

Key:

1. Background division; 2. second division front plan; 3. first division of front plan; 4. assignment

In each of the three divisions the system insures the possibility of package processing of assignments, and for each of the divisions the input flow of assignment itself must be prepared. In the initial stay, that is, after execution of the initial loading procedure, the system is tuned to reception of the flow of assignments in the background division. The connection of the input flows of assignments to the system in the divisions of the front plan is realized by the operator. In Fig 4 we have the diagram of the input flows of assignments in the case of multiprogram processing.

The assignment to execute the background program is accepted by the assignment control program from the input flow of the background division. Independently of the user's desire, the system always adjusts to the execution of package assignments in the background division, and itself demands the arrival of the next assignment after completion of the preceding assignment. The operator cannot cancel the package processing in the background division.

The assignment for the execution of the program of the front plan is accepted by the Assignment Control program from the input flow of the corresponding division after the operator has given instructions to the system to begin package processing in this division from the panel typewriter. After starting the processing, the system itself will require the arrival of the next assignment until the operator cancels the package processing in the given division. After cancelling the package processing, the given division of the front plan becomes free.

The time distribution of the processor among the programs being introduced simultaneously is supervised by the Supervisor, using the priorities established for the program. The background program is always assigned the lowest priority, the priority of the program in the second division of the front plan is higher than the priority of the background program, the program of the first division of the front plan has the highest priority. The priority determines the degree of urgency of the program and is perceived by the Supervisor as an indication of the sequence for execution of programs simultaneously in the basic memory. The standard operating cycle of the Supervisor looks like the following: the Supervisor takes control on the interrupt signal, processes this signal and ends its operation by transfer of control to the problem program having the highest priority among the programs ready for operation (Fig 5). The program with highest priority is free of control when a cause arises preventing continuation of the processing. The program with lower priority will be halted as soon as obstacles to the continuation of the program for the division with higher priority disappear. The background programs and the programs of the front plan are included in the processing and end asynchronously.

Since the priorities are attached to the divisions, the effect achieved during multiprogram processing essentially depends on the mixture of simultaneously executed programs.

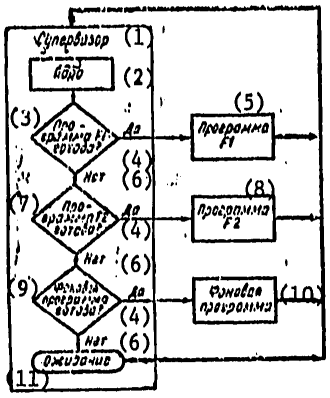

Figure 5. Preparation of the Supervisor during multiprogram processing;
F1 -- first division of the front plan; F2 -- second division of the front plan; → -- program transfer; ⇒ -- transfer by interrupt signal

Key:

1. Supervisor
2. Nucleus
3. Program F1 ready?
4. Yes
5. Program F1
6. No
7. Program F2 ready?
8. Program F2
9. Background program ready?
10. Background program
11. Waiting

Communication with the Operator

The DOS YeS operating system grants the operator greater possibilities for control of the computing process. The operator receives information from the system which characterizes the state of the system and the executed programs. Having this information, the operator can make substantiated decisions and control the operation of the system, giving directives to it.

Most frequently the panel typewriter is used for communication with the operator. Instead of the panel typewriter the system permits the use of an alphanumeric printer. However, in this case the possibilities of the operator and the system are limited (for example, without the panel typewriter the multiprogramming processing mode is impossible). Hereafter it is assumed that the panel typewriter is used as the communications device.

The messages coming from the programs to the operator can be of an information nature and be of a question nature. For each question of the system, the operator is obligated to give an answer, assuming the solution on the basis of analyzing the operating situation and give the system information required by it to continue working. In order to facilitate the work of the

operator, the DOS YeS messages have a standard structure, they are operated and equipped with an explanatory text. The operator answers also have standard form and are required in cases where the system permits several versions of continuation of the work. In such cases the operator must select and indicate for the system the specific version of the continuation. Usually the answer of the operator to the system is short, is made up of one word, for example: RETRY, IGNORE or CANCEL.

The operator is able to influence the calculation process, giving directives to the system when necessary. Thus, by using the directives the operator can stop the execution of the current assignment, halt the computation process in any of the divisions after completion of the assignment or the assignment step for execution of certain operations (for example, with respect to servicing the peripheral devices).

One of the constant functions of the operator on the computer is the preparation and servicing of the peripheral devices: the changing of the packages of discs, the loading of the magnetic tape reels, loading and unloading the punch cards, and so on. Sometimes operator intervention is needed if an error occurs in the peripheral device. If an error occurs during input-output, the system, as a rule, halts the execution of only one program among the ones being executed simultaneously. The program halts in the division where the device is attached at the given time, during operation of which an error is detected. The execution of the halted program can be resumed after the operator performs defined operations on the device. When the device is ready, the halted input-output operation is started up automatically. In the majority of cases in order to renew the execution of the halted program the response of the operator from the typewriter is not needed.

The role and the responsibility of the operator increase when the system functions in the multiprogram mode, and the operator must see to each of the programs being executed simultaneously. In order that the operator be able to determine in which division the program requires attention, the system flags each communication printed out on the panel typewriter with an identifier of the division that sent the message. The presence of the division identifier and the message is necessary, for the messages of all of the programs operating simultaneously are printed on the same panel typewriter, and their sequence cannot be determined in advance. Thus, after a message from the background program, the message can be printed out pertaining to any of the divisions in the front plan. If it is necessary to follow the course of the execution, for example, of the background program by the messages, then the operator must select messages only from the background in the typewriter output. The presence of the division identifiers in the message facilitates this procedure for the operator.

Supervisor

The existence of the supervisors arises from the necessity for processing interrupt signals and the desire to organize multiprogram processing with

more efficient use of the machine resources. In the case of multiprogram processing the central processor is switched from one program to another, executing all of the operations together and not in series. The switching of the processor is realized by the supervisor, using its program selection mechanism encompassing the class of problems such as priorities, loading of programs, memory distribution, distribution of the peripheral devices and control of their operation. Among the other properties of the supervisor which can be used by the problem programs, it is necessary to know the organization of communications with the operator, the restoration of the functioning in the case of errors in the peripheral devices, servicing of the timer, the creation of check points, the completion of the execution of the programs, and the recording of system operations.

Fig 6 shows the composition of the DOS YeS Supervisor.

A distinguishing feature of the Supervisor in the DOS YeS system is the modularity which permits the user to select only the needed functions and capabilities which can be realized by the Supervisor from the large number available and to generate its own supervisor realizing these functions and capabilities. The generation of the required Supervisor permits space to be saved in the basic memory and the time required for operation of the Supervisor to be shortened during the functioning of the system.

To generate a specific Supervisor realizing only selected functions means to create the required, specific nucleus of the Supervisor. For this purpose the user must determine which properties (among those available in the DOS YeS) he would like to include in the Supervisor. This determination is made by assigning the corresponding generation parameters. The size of the Supervisor nucleus essentially depends on the properties required by the user. For example, the inclusion of the multiprogram processing properties in the Supervisor nucleus increases its size by 1384 bytes, and the monitor zone must be enlarge accordingly.

Simultaneous Execution of Programs. The simultaneous execution of several programs is realized in the DOS YeS only if this possibility is included in the Supervisor at generation time. The simultaneously executed programs must be in the basic memory, each in its own division. The division in which the program is executed is called active. If there is no program in the division or no division in the system, then the division is considered inactive. The background division begins always after the monitor zone; it must be 10K bytes or more than 10K bytes, which makes it possible to increase the speed of execution of the division programs, including such frequently used programs as the Assembler and the Editor.

As a result of the background division, the entire basic memory can be distributed between two divisions of the front plan. The size of each division and also its initial address must be multiples of 2K. This requirement arises from the method of organizing the basic memory protection which is adopted in the YeS EVM computers and is provided for in the DOS YeS.

For organization of protection, the entire basic memory is provisionally divided into blocks of 2K bytes each. The blocks allocated to one division receive the same protection switch. The monitor zone has a protection switch equal to zero; the background division, equal to one, the second division of the front plan, equal to two, and the first division of the front plan, three. During operation the Supervisor can see that the program cannot reference the basic memory zone which is used by another program. In case of an erroneous reference, the YeS EVM [unified computer system] hardware generates the "memory protect" interrupt signal which is processed by the Supervisor.

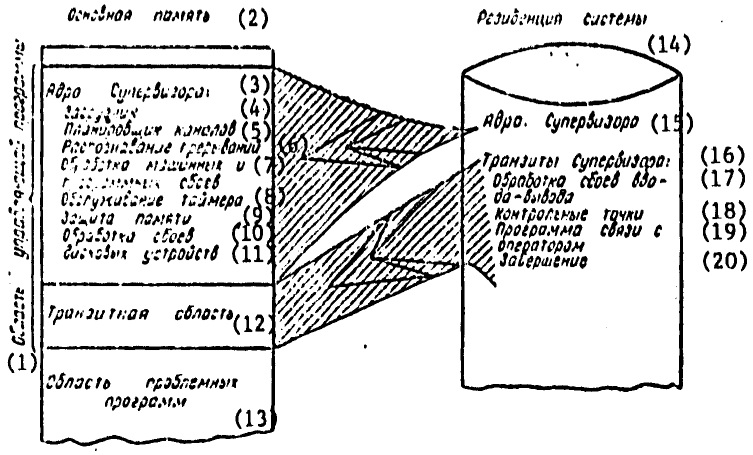

Figure 6. Composition of the Supervisor

Key:

1. Monitor zone
2. Basic memory
3. Nucleus of the Supervisor:
4. Loader
5. Channel planner
6. Recognition of interrupts
7. Processing of machine and program failures
8. Servicing of the timer
9. Memory protection
10. Processing of failures
11. Disc units
12. Transit zone
13. Problem program zone
14. System residence
15. Nucleus of the Supervisor
16. Supervisor transits:
17. Processing input-output failures
18. Check points
19. Program for communications with the operator
20. Completion

The provision of memory protection in the DOS YeS is a property which is included in the system if the user desires it when generating the Supervisor. If in the specific system there is no protection, then the "memory protect" interrupt signal is ignored. The requirement that the Supervisor react to the "memory protect" signal is a necessary requirement for the multiprogram

processing mode. Therefore on generation of the Supervisor with multiprogram processing, the memory protect mode must also be requested.

During the system generation time, the distribution of the basic memory among the divisions can be indicated. Then during functioning of the system, the operator can change the interfaces. In this case the redistribution of the basic memory between the divisions, as a rule, is executed when the front plan divisions are inactive.

In the case of simultaneous execution of several programs the machine resources are used more completely than when executing only one program. This is achieved as a result of the fact that there are a large number of programs in which the processor actually does not work much of the time but only waits for completion of the input-output operations. The forced idle time of the processing can be used by another program. Any time when the program is forced to wait for completion of the input-output operations in one division, the Supervisor transfers control to the other division, the program of which is ready for execution in accordance with the priority attached to it. The first division of the forward plan has the highest priority, and therefore always when the program of this division is ready for execution the Supervisor transfers control to it, interrupting the program of any other division. Thus, the program with the highest priority is always executed when it is ready for execution. With this problem selection principle, there is a probability that the program with high priority will be executed alone, and the Supervisor cannot transfer control to other programs. This actually can occur if the program with high priority has very few references to the input-output devices.

For purposes of maximum utilization of the computer resources, it is necessary specially to plan the mixture of simultaneously executed programs. The mixture must have programs present in which a significant number of input-output operations on the devices with low speed are used (the input and output of punch cards, printing, and so on). These programs create a heavy load on the input-output devices, but they contain a small number of internal processing operations, and they do not make full use of the simple processor of the computer. In the mixture there must also be a background program of computation nature with a significant number of processing operations and with small load on the input-output devices.

The effect which can be achieved from the successfully selected mixture of programs depends on the priority attached to the programs. The programs requiring a significant number of input-output operations must be executed as programs with high priority. The programs with a large volume of internal processing operations must be made with low priority, for otherwise they can lower the carrying capacity by monopolizing the time of the central processor for a long period.

Starting Up the Input-Output Operations. The operating procedure used in the DOS YeS system with peripheral devices presupposes that the information

about the idle time of the processor is communicated to the Supervisor by the program itself as a result of the fact that the program expects completion of the input-output operation.

In order to be able to execute any input-output operation, a channel program [driver] must be prepared for it. It is made up of one or several input-output instructions describing the required input-output operations. The channel program is part of the problem program itself. Always when it is necessary to perform an input-output operation the program communicates with the Supervisor on the necessity to start the channel program for this operation. After receiving this request, the Supervisor proceeds with processing of the request and determines the specific device for which it has been made. If at the given time this device is busy with the execution of a previously started operation, then the new request is stored by the Supervisor; a queue is formed for the device. If the device is free and ready to operate, the Supervisor immediately starts the requested input-output operation, transmitting information about this operation to the device. In this case the start-up operation ends independently of whether the operation has been started or it has been put in the queue, and the Supervisor transfers control to the problem program. If only one problem program is being executed, the control is returned to it; if there are several programs being executed, the control is transferred to the program having the highest priority and ready to operate.

The input-output operations placed in the queue by the Supervisor are started one after the other in order of the queue. No additional requests to start the input-output operation are necessary from the program.

Thus, after starting the input-output operation, the processor and the device receiving the required information can operate simultaneously. The program which has put out the input-output request waits for completion of the operation, and at that time the program of another division is executed.

The interrupt signals occurring during the operation of the peripheral device inform the Supervisor about the course of execution of the input-output operation. The Supervisor processes the interrupt signals caused by freeing up of the channel, the control circuit, the input-output device, by pressing the "Attention" button and also signals caused by detected errors (data errors, failure of the device, the device not ready for operation, and so on). On the occurrence of any interrupt signal, the execution of the program is interrupted in any division, and control is transferred to the Supervisor. The Supervisor undertakes the specific operations on each of the signals.

The interrupt signals caused by freeing of the channel, the control unit and the input-output device indicate completion of operation of the corresponding equipment on the performance of the operation. If the Supervisor has established that the operation has been successfully completed, it excludes the request for this operation from the queue, it

investigates the possibility of starting up the next operation from the queue to this device and under favorable conditions realizes startup of the next operation. The described operating cycle of the Supervisor is repeated for each input-output operation. An example of simultaneous execution of the programs in three divisions as a function of the request for input-output is presented in Fig 7.

Each input-output device, the operation with which is provided for by the Supervisor, must be indicated at the system generation time. The Supervisor services only the devices which the user has indicated during the creation of his specific nucleus. If the device has not indicated the Supervisor, then the system considers that it is absent in the computer and, consequently, no work with it is provided for.

All the operations connected with starting up the input-output operations and also processing the input-output interrupt signals are performed by part of the Supervisor which is called the channel planner. The channel planner always enters into the Supervisor nucleus. However, this does not mean that the channel planner is invariant in all versions of the Supervisor nucleus. It is possible to indicate the properties of the channel planner which are the generation parameters, and they are included in it depending on what the user wants, for example:

Organization of the simultaneous execution of the input instructions for the disc memories;

Protection of the files on the disc;

Use of the disc for systems logical circuits;

The buffering regime for messages to the operator.

The inclusion of any of the indicated properties in the composition of the planner implies enlargement of the Supervisor nucleus. Therefore the user is offered the possibility of choosing the following: having the smallest Supervisor nucleus with limited properties or obtaining the best characteristics during operating time as a result of enlarging the nucleus.

Organization of Simultaneous Execution of the Feed Instructions for the Disc Memories. The channel program for exchanging information with the disc contains feed instructions which set the read-write heads on the outside track of the disc. After starting the channel program for the disc units the signal that the channel is free and the device is free will be received by the Supervisor only after completion of the execution of all instructions of the channel program. All of this time the Supervisor cannot start up the input-output operation with any disc unit connected to the same channel. This means that until the channel becomes free, all the programs will be waiting which require execution of the input-output operations for disc units connected to this channel. The channel equipment will permit execution of feed instructions simultaneously on all the

devices connected to the same channel. Out of the total time spent by the channel on execution of the channel program, a significant part goes to the feed instructions, and it would be possible to use this time for other operations.

On assigning the corresponding generation parameter, the operating system allows the user the possibility of obtaining a Supervisor which will organize the simultaneous execution of the feed instructions on the disc. In this Supervisor the channel planner isolates the feed instruction from the channel program, starts it and while it is being executed, transfers control to another program which, in turn, can send a request to execute an input-output operation for another disc unit. When processing this new request the Supervisor also isolates the feed instruction from the driver and starts it, that is, two programs are serviced simultaneously. As soon as the feed instruction is executed, the execution of the remaining part of the driver continues. Thus, an overall increase in output capacity of the system can be achieved as a result of more efficient use of the computer equipment. This is especially significant for computers having a large number of disc units connected to one channel.

The channel planner will isolate the feed instruction from the driver only if this property was requested during the generation of the Supervisor nucleus. Otherwise the feed instructions in the drivers for reference to the disc units will take the channel for its entire execution time.

Protection of Files on Discs. The operating system permits various problem programs to use nonoverlapping sections on the same package of discs. These programs can be executed simultaneously, and then the problem arises of organization of the protection of the files from destruction by foreign programs.

In the DOS YeS, the file protection on the discs is organized as follows: the information about the boundaries of the various sections of the discs allocated to each problem program is stored in the Supervisor tables. On any reference to the discs, the Supervisor checks the correctness of the reference -- the problem program is permitted to reference only its own sections. The program cannot write information in a foreign file and read foreign information. if the program tries to reference a section of the disc that it does not belong to, the Supervisor halts the execution of this program and tells the operator about it.

The file protection mode for the disc is provided for by the Supervisor only if this property is requested during generation time.

Use of Discs for System Logic. The method of logic units permits programs to be created which are tied only to the type of peripheral device and do not depend on the address of the specific physical input-output unit.

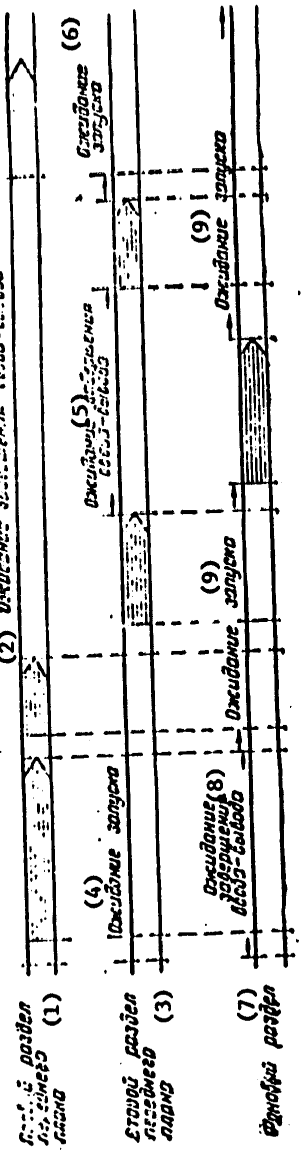

Figure 7. Example of the simultaneous execution of programs in three divisions

Key:
1. First division of the front plan
2. Waiting for completion of input-output
3. Second division of the front plan
4. Waiting to start
5. Waiting for completion of input-output
6. Waiting to start
7. Background division
8. Waiting to complete input-output
9. Waiting to start
10. Supervisor
11. Processor
12. Input-output interrupt

For example, if the program plans to take data off punch cards, it must take into account the specific nature of the operation of the punch card input unit, but it is indifferent to the program from which of the punch card input units available to the computer the data input will actually occur. At the same time, it is impossible to use another type of device, for example, a magnetic tape for data input in place of the planned punch card input unit.

For system logic such as the systems printer, the system data and assignment input and the system punch, the system provides further possibilities. The system programs in which these logic units are used do not depend on the type of physical devices. For each system logic there is a possibility of selecting a specific type of physical device from among the available ones. This can be either specific devices (punch card input and output, printer) or magnetic tapes, or discs. The possibility of selecting the type of peripheral device in each specific assignment permits efficient use of the system resources and increases the effectiveness of its functioning.

The discs can be used as the enumerated system devices only if this property of the system has been requested during the generation of the Supervisor.

If this property has been requested, then parts are added to the channel planner which provide for monitoring the correctness of the use of the system files.

Independently of whether the disc file protection mode is requested or not, the system files on the discs are protected.

Buffering of the Operator Messages. All of the programs of the operating system generate messages to the operator on the panel typewriter. Any user program can also output messages to the operator. Since there is only one panel typewriter in the system, and it is a slow unit at that (100 to 150 milliseconds is needed for each printed character), the time for execution of the driver to output the message to the typewriter is comparatively large -- up to several seconds. When there are frequent messages to the operator, the typewriter will become a brake which slows the program execution speed.

In order to output messages to the operator on the panel typewriter the problem program has a special subroutine. The overall functioning flow chart for this subroutine is as follows: the text of the message is placed in the output buffer and a request is sent to the Supervisor to output it from this region. At any time that a new message is to be output, it is necessary first to see that the preceding message has already been printed out and the output buffer is free. Only then can a new message be entered. If the preceding message has still not been printed out, then the output buffer is busy, and the subroutine cannot prepare the next message; it is forced to wait for the typewriter to be released, and the entire problem program is idle for all of this time. The idle time can be great, especially in the multiprogramming mode, for any of the simultaneously

executed programs can send a request for the typewriter to the Supervisor, and a queue is formed in the Supervisor for this unit.

In this case an increase in output capacity of the system can be achieved only when the problem program is executed simultaneously with printing out the messages. For this purpose it is necessary that the problem program be able to occupy its message buffer zone until the message actually is output to the typewriter.

The transmission of the text of the message to the operator from the buffer of the problem program to another basic memory buffer would be a solution to this problem. Since the input-output operations in the DOS YeS system are realized by the Supervisor, it is natural that this buffer zone must belong to the Supervisor, and the Supervisor can send a message text to it. Thus, on receiving a request to output the message to the typewriter the Supervisor must transfer the message text from the problem program buffer to its own buffer and then start the message output operation for the message in the buffer. The time for direct printing of the message on the typewriter will be made compatible with further execution of this problem program. This is especially significant for programs which frequently generate messages tothe typewriter.

The property of the Supervisor to accept messages output to the typewriter in its own buffer is called bufferization of the messages to the operator and is a generation parameter. If the bufferization mode for the operator messages is required during generation of the Supervisor, then the information to be output to the typewriter is first sent by the channel planner from the problem program buffer to the Supervisor buffer before beginning output. The message will be output to the typewriter from this buffer. After transferring the message to the buffer the problem program receives word that the operation of outputing the message to the typewriter has been completed, and it can be executed farther just as if its message had been output to the typewriter.

During generation the user can indicate what number of buffers must be reserved in its specific Supervisor. The Supervisor can accept up to nine messages in its buffers (the maximum number of buffers is nine), put them in a queue and then output them one after the other to the typewriter.

The presented examples do not exhaust all the properties of the channel planner which are defined by the generation parameter. They are selected for illustration of how the user can expand the properties of his specific Supervisor. There are a number of generation parameters which influence the operating algorithms of the Supervisor itself, optimizing one function of the Supervisor or another with respect to execution time. Some of them will be discussed further.

Input-Output Error Processing. The Supervisor executes many functions connected with the organization of input-output during the normal course of this operation: the keeping of queues for the devices, starting the

Input-output operations, processing the interrupt signals on completion of input-output. However, the input-output operation is not always executed correctly -- errors can be detected during the execution process, from easily correctable ones to disastrous ones. Even if the occurrence of errors has low probability, the operating system must provide for the possibility of their appearance and the taking of the required measures.

In the YeS EVM computers on execution of the input-output operations, deviations from the normal execution of the operation are detected such as the improper length of the data, a special case in the device, various failures in the channel or the device. Improper length of the data, for example, when executing a punch card input operation means that the number of symbols input from the punch card is less than or greater than 80. A special case, for example, in the structure of the input from punch cards occurs if there are no punch cards in the feed hopper of the reader.

Such deviations as improper length of the data and the special case in the devices do not indicate incorrect execution of the operation, and they are perceived by the Supervisor as normal. The Supervisor transfers the information about this to the problem program, and the program itself must decide its future course of action, for it is only in the problem program that the information about admissible or inadmissible deviations from the norm can be available. Thus, in the program it can be taken into account that information is prepared on the cards in only 30 columns and not in all 80.

Special attention is given to situations connected with the appearance of failures during execution of the input-output operations. Each device and each operation executed by the device are controlled on their own, and the reaction of the device to the errors is specific in many cases. This, in turn, requires that each have its own procedures for elimination of the consequences of an error. The desire to keep the programmer from having to develop procedures to avoid input-output errors and include them in his own program has led to the fact that these functions have been transferred to the Supervisor.

The actions taken by the Supervisor in case of the appearance of input-output failures are differentiated with respect to types of devices and types of failures. Here such factors as the specific nature of the peripheral device, the form of operation, the state of equipment at the error detection time, the procedure for processing failures required by the problem program and also the degree of completeness of the actions of the Supervisor itself in each specific case are taken into account.

All of this information offers the Supervisor the possibility of unique determination of what its reaction should be to each specific failure. The Supervisor can take one of the following actions:

It can try automatically or with the help of the operator to eliminate the failure;

It can ignore the failure;

It can return control to the problem program on request;

It can stop the execution of the assignment.

An effort is made by the Supervisor to eliminate the failure without operator intervention for such devices as the magnetic tape or disc. For these devices the Supervisor repeats the execution of the operation in which the failure was detected. In the case of successful repetition the operation is considered executed correctly, and the overall operation of the program continues just as if no failure had occurred. Only if errors occur on each of the repetitions (the total number of which is fixed by the system) will the Supervisor communicate with the operator about the presence of a failure in a specific device. Then immediate dumping of the program can follow if the Supervisor has established that no further actions of the system or the operator can eliminate the error or additional action on the part of the operator can be requested.

For certain devices and types of errors the Supervisor can eliminate the consequences of a failure only after intervention of the operator in the operation of the device. For example, if a punch card has jammed in the input unit, the Supervisor reports an error to the operator. The operator must replace the jammed punch card, and then ready the device for operation. After this, the punch card input operation is again executed, and the execution of the program continues.

The actions under taken by the Supervisor do not always meet with success, and then the Supervisor prints out a message about the error on the panel typewriter and, as a rule, requires that the operator respond how the operation is to be continued. In the general case the operator can instruct the Supervisor in one of the following versions of continuation:

Again repeat the actions to eliminate the error;

Ignore the error and continue to execute the program;

Halt the execution of the assignment and dump the program.

Among the input-output errors, the Supervisor isolates the errors which do not lead to assertion of the information and have no influence on the course of execution of the program. For example, after completion of the operation of rewinding the magnetic tape, the Supervisor discovers that the device has established a data failure. However, during the rewind no data are transmitted which means the data failure has no bearing on the operation. This failure and similar ones are ignored by the Supervisor.

The standard actions taken by the Supervisor in the case of occurrence of input-output errors can fail to satisfy the requirements of the specific program to one degree or another. The specific program is a condition to have significantly broader information about the situation that has developed during the process of its execution which on occurrence of erroneous situations will permit the program to find better methods of continuation. In other words, the problem program can have its own procedures for eliminating errors, and in this case the mission of the Supervisor is to put these procedures into operation when an error occurs. The Supervisor allows the programmer the possibility of requesting the indication of the presence in its program of its own procedures for processing failures of specific devices. In this case it transmits all the information about the failure that has occurred to the program.

The occurrence of errors during the execution of input-output operations is possible, but they are rare. At the same time the number of possible errors and the actions taken with respect to each of them are quite high and consequently the programs realizing these procedures have large volume. Therefore the procedures for eliminating failures are partially found in the nucleus of the Supervisor, but they are transit programs in the basic pass of it. The Supervisor nucleus only contains the programs for processing the failures of the disc units, for not only the user programs constantly reference these units, but also the DOS YeS system itself.

The programs realizing the above-described operations with respect to processing input-output errors are included in any specific Supervisor, that is, they do not depend on the parameters of the generation of the Supervisor nucleus.

In addition, two possibilities are connected with the input-output error processing procedures. These two possibilities are included in the Supervisor as the user desires depending on the generation parameters. This includes obtaining statistics on the failures of the magnetic tape and a reduction in time spent on repetition of the input-output operation.

The keeping of the statistics about the failures of the magnetic tape consists in the fact that during execution of the assignment the Supervisor accumulates information about the number of failures of different types for each magnetic tape. On completion of the execution of the assignment, a message is printed on the panel typewriter about the number and types of detected magnetic tape failures.

Another capability increases the efficiency of the system in that it permits a decrease in the time spent by the Supervisor on repeating the input-output operation. When there are input-output failures in the cases where the operation must be repeated, usually the execution of the entire driver is repeated and not only the instruction during the execution of which the error occurred. It is possible to reduce the time spent on eliminating an error if the entire driver is not repeated from the beginning

but, if possible only part of it beginning with the instruction triggering the failure. The programmer himself determines whether it is possible to eliminate the error in his program without repeating the entire driver as a whole, and he reports this to the Supervisor. If this property is not requested during generation of the Supervisor nucleus, then on repetition of the input-output operation the entire driver is executed.

In addition to the errors in the input-output units, during operation hardware and software failures can be detected which are also processed by the Supervisor.

Processing Program Failures. Program failures occur if there are errors in the program: an inadmissible operation code, a nonexistent memory address, disturbance of the memory protection, and also on occurrence of extraordinary situations with the data: overflow during arithmetic operations, an attempt to divide by zero, loss of significance or disappearance of order of magnitude in floating-point operations, and so on.

A failure caused by a program error is perceived by the Supervisor as a situation which it cannot correct. The role of the Supervisor in this situation reduces to most precise localization of the error and establishment of its nature, communication with the operator about this and halting the execution of the assignment. The message about the error is always output in standard form to the panel typewriter. In addition, the programmer can have a printed copy of memory when the program failure was detected. On analyzing this program dump, the programmer can establish the cause of occurrence of the error in this program.

A characteristic feature of program interrupt is the fact that some of them can be masked by the user program. The program failure for which the interrupt is masked will be ignored by the system, that is, the system does not print out information about the failure and does not halt the execution of the assignment.

In the standard case a program failure causes halting of the execution of the assignment. It is possible that this outcome will not be satisfactory to the programmer in a specific problem program. It is possible in the program to provide in advance for the appearance of certain errors and to take measures to find a means of continuing the program. The program could also be executed after the occurrence of a program failure in it if the program has included a subroutine in it for a nonstandard reaction to program failures, for example, to overflow.

In order to be able to make use of one's own program failure processing procedures, the corresponding condition must be set up:

The Supervisor must know that the problem program will process the program failures occurring in it;

The Supervisor must know how to activate these procedures at the required points in time.

The problem program itself communicates to the Supervisor that it has a subroutine for processing the program failures and indicates the location of the subroutine in memory. The program itself also establishes the limits within which its own program failure processing procedures must operate. Within the established limits the standard procedures of the Supervisor while processing program failures are inactivated (Fig 8).

The possibility of using one's own subroutine to process program failures is provided for during generation of the Supervisor, and it is a nucleus generation parameter. If this property is not requested during generation of the Supervisor, then only the standard reaction of the Supervisor to the program failures will be admissible.

Processing Machine Failures. An interrupt for machine failure indicates that incorrect functioning of the equipment, for example, the basic memory or channel, has been detected. Usually this implies calling on the engineering personnel for repairs. On receiving the interrupt caused by the machine failure, the Supervisor puts the computer in the "serious halt" condition: any operation ceases, and all interrupts are forbidden. After elimination of the failure, the operation of the computer must begin with the initial loading procedure.

System Loader. One of the functions of the Supervisor is loading the programs into the basic memory for execution.

In the DOS YeS system, all the programs are loaded into the basic memory by the part of the Supervisor which is called the system loader. The system loader is part of the nucleus of the Supervisor and always functions when any program is to be loaded into the basic memory, whether it is the transits of the Supervisor, any other programs of the operating system or programs prepared by the user. The procedure for loading the programs provides for solving such problems as finding the program in the external memory, determination of the location in the basic memory where the program is to be loaded, and determination of the point of entry into the loaded program.

In the DOS YeS system the only place where the programs ready for execution can be located and where they are loaded in the basic memory is the library of absolute modules. The programs are entered in the library of absolute modules by the Editor in the program editing stage.

The programs ready for execution constitute one program phase if the program has simple structure or several program phases if the program has a structure with overlay. The program phases in the library of absolute modules are stored exactly in the form they will take in the basic memory after loading. Each program phase is always adjusted to a specific field

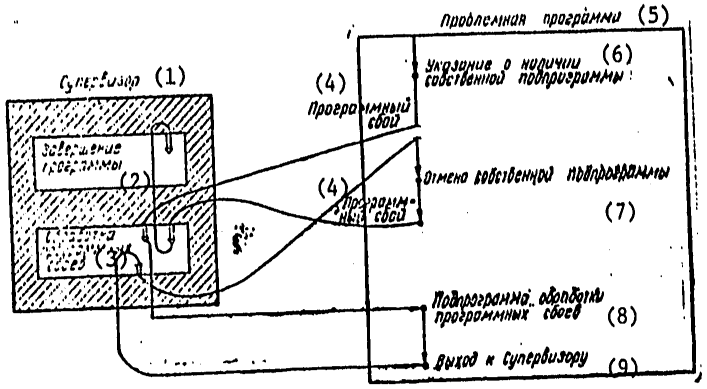

Figure 8. Processing interrupts for program failures

Key:
1. Supervisor
2. Completion of program
3. Processing of program failures
4. Program failure
5. Problem program
6. Indication of the presence of its own subroutine
7. Cancellation of its own subroutine
8. Subroutine for processing program failures
9. Exit to the Supervisor

of the basic memory, and can be called for execution only in this field. The loading address for the phase is the address of the beginning of the field of the basic memory in which the phase will be loaded for execution. This loading address is determined in the editing stage, it is stored in the library of absolute modules, and it is used by the System Loader any time the phase is loaded into the basic memory.

In the library of absolute modules, in addition to the text of the phase and its loading address, the name of the phase and the point of entry into it are stored. The phase name is the only attribute for the System Loader by which it finds the phase in the library of absolute modules. Therefore, whenever the phase is to be called into the basic memory, the System Loader must be given the name of this phase.

All of the auxiliary information about the phase (the name of the phase, the load address, the point of entry to it) and, in addition, the phase address in the library of absolute modules are stored in the table of contents of the library of absolute modules. This table of contents is examined by the System Loader every time a phase is to be found in the library of absolute modules and loaded into basic memory. After the phase with the required name has been found in the table of contents, the System

Loader calls it out of the library into the basic memory. This operating algorithm of the System Loader permits the time spent by the system on finding the phase in the library to be decreased, for the table of contents of the library is significantly smaller than the library itself.

Thus, the program is ready for execution and is located in the library of absolute modules. How does the Supervisor determine when and what phase of the program must be loaded into the basic memory? By its initiative, the Supervisor loads only its own transits and the assignment control program into the basic memory. The requests to load any other programs of the operating system and user programs are sent to the Supervisor by the Assignment Control program and the problem programs themselves.

The Assignment Control program determines from the assignment itself the name of the program, the execution of which is required in the assignment. The Assignment Control program indicates this name in the request which it addresses to the System Loader. After receiving the request to load, the Supervisor finds the required phase in the library of absolute modules and loads it in the basic memory in accordance with its load address. Then the phase is activated, and takes control immediately after loading if this is the only program in the computer. If several programs are being executed simultaneously on the computer, the phase will receive control in accordance with the division priority in which it is loaded.

After the loaded program has been executed, the Supervisor, on its own initiative, calls the Assignment Control for taking the next assignment.

If the program has a structure with overlay, then the Assignment Control generates a request to the Supervisor to load only the basic (first) phase of the program. The request to load the remaining phases must be generated by the program itself. Depending on the type of request, two cases are distinguished: after loading control is transferred to the loaded phase or the phase making the request continues to be executed.

When making the request to load, in addition to the name of the phase, the programmer can give the loading address and the point of entry. If this additional information is assigned, it is given priority over that information which accompanies the phase in the library of absolute modules.

The System Loader never processes the text of the loaded program. It only copies the program stored on the disc into the basic memory.

During operation on a computer in the multiprogram mode, the user can need to have the same program executed in different divisions. In the general case whenever the program is to be executed in a division other than that for which it was prepared, this program must again be edited. At the same time the addressing procedure used in the YeS EVM computers makes it possible for the programmer to create a program which will be invariant with respect to the location of it in the basic memory. In the YeS EVM computers, the

addressing procedure is such that the basic part of the program text will not change when the program is moved; only the address constants and the input-output instructions depend on the load address, the number of which in the program is not so great. The programmer can include instructions in the text of the program which corrects the address constants and the input-output instructions of the program itself during execution in accordance with the loading address.

The program written in this way can be loaded in any division of memory: it does not depend on the specific loading address. In the DOS YeS system, such programs are called self-displacing. For determinacy, in the editing phase of the self-displacing program, a loading address of zero is formed, and by this attribute the Supervisor can distinguish the self-displacing programs from the nonself-displacing ones. When a request comes in to load self-displacing programs, it is loaded in the division from which the request to load it was made. During execution the program itself can generate a request to load other of its phases which also must be self-displacing. The loading address must be indicated in the request.

The System Loader is a part of the Supervisor nucleus, for it is used quite frequently both by the Supervisor itself to call its own transits and the silent control program and by all the remaining programs. The described functions of the System Loader do not depend on the generation parameters and are provided for in any specific Supervisor. Nevertheless, one of the nucleus generation parameters has a direct effect on the operation of the System Loader. During the loading process the System Loader references the discs: it reads the table of contents of the library of absolute modules from the disc and also the requested phase from the library itself. Just as any program, the System Loader is obligated to wait for completion of execution of the read operations from the disc which it has requested. During this waiting when operating in the multiprogram load, problem programs could be executed. However, this capability is provided for only if it is requested at the time of generating the nucleus. If this property has been requested, then a special mode is set up for the execution of the input-output operations requested by the Supervisor itself during which the time that the Supervisor waits for completion of its own input-output operations can be used for execution of problem programs. Otherwise, the computer is idle for all of this waiting time.

This property improves not only the operating algorithm of the System Loader, but also other parts of the Supervisor which request the execution of input-output operations, for example, the transits for processing the input-output errors.

Program for Communication with the Operator. An important role in the Supervisor is played by the programs which communicate with the operator and permit the operator to influence the computation process. This possibility is made available to the operator by the programs available in the Supervisor which are called Attention programs. During operation of the

system, the times at which the operator will intervene in the operation of the system are not determined either in time or with respect to the place in any of the programs. These events can be determined only by the operator, and the duty of the system in this case is to give attention to the operator any time he requests it.

In the DOS YeS system when the operator has determined that he must give an instruction to the system, he reports this to the Supervisor through the panel typewriter. The operator presses the "Attention" button on the panel typewriter, and this signal is picked up by the Supervisor as an indication that the operator requires transfer of control to him. The Supervisor informs the operator of readiness to take his instructions, sending the corresponding message to the typewriter, after which the operator can input his instructions from the typewriter which are then received and processed by the Attention program. After all of the directives planned in the given control sequence have been entered, the operator reports the end of the communications session to the system.

The most important of the functions executed by the operator in controlling the DOS YeS system are the following:

Starting up operations in the divisions of the front plan;

Redistribution of the basic memory among the divisions;

Halting the execution of the assignment in any of the divisions;

Halting the package processing of the assignments, requesting a break after completion of execution of the current assignment or assignment step in order to give additional instructions for the next assignment during this halt time or to service a peripheral device;

Indicate which of the divisions the timer can use.

The operator can request that the system give him detailed information about the executed assignments, the distribution of the basic memory among the divisions. The list of directives of the Attention program is presented in Table 1.

The functions provided for by the Attention program do not include the entire list of operator possibilities in the DOS YeS system. The Attention program only realizes the functions which are of a universal nature, are applicable for the program of any division and the execution of which can be requested at any point in time. In addition to these universal control capabilities the system grants the operator other possibilities which permit him to participate in preparation of the assignments for execution on the computer. For example, in the interval between assignments the operator can reassign specific physical devices for the logic used in the assignments, request the execution of control operations for the magnetic

tape, forbid the use of a specific input-output device. This type of capability is realized not by the Supervisor, but by the Assignment Control program, for the necessity for these capabilities arises, as a rule, between assignments, and it is during this time that the Assignment Control program functions in the system. More details will be given about this later.

Table 1

List of Attention Program Directives

Directive	Function
ALLO	Permits distribution of the basic memory among the divisions
MAP	Causes printout of information on the distribution of the basic memory among divisions on the typewriter
CANCEL	Permits removal of the assignment, that is, halting the execution of the current assignment
BATCH	Permits the beginning of package processing in the division of the front plan or continuation of the execution of an operation halted by the operator
START	Permits resumption of execution of an operation halted by the operator.
TIMER	Permits indication of the division in which the timer will be used to control the time interval
MSG	Permits activation of the subroutine for communication with the operator in the problem program being executed in the front plan division
PAUSE	Requests a pause, that is, permits indication to the system of the necessity to halt the processing of the input flow of the assignments after completion of the next step of the assignment
LOG	Permits a request to print out the controlling operators and directives of the input flow of the assignments on the typewriter
NOLOG	Permits cancellation of the printing of the controlling operators and directives of the input flow of assignments on the typewriter

The possibilities which are made available to the operator by the Supervisor and Assignment Control program can be insufficient for the specific problem program. During the process of execution of the problem program the operator can execute nonstandard functions specific only to this program, including with respect to controlling the course of execution of this program. All of the nonstandard functions of the operator must be realized by the problem program itself. The Supervisor must know about the presence in the problem program of the program for communication with the operator itself, and the Supervisor must know how to transfer control to it to get around its Attention program.

This property which permits the problem program to use its own subroutines for communication with the operator is a Supervisor generation parameter. If the Supervisor has this property, then each of the simultaneously executed problem programs can have its own subroutine for communication with the operator. If this mode is not requested, then the operator can use only the standard directives of the Supervisor and the Assignment Control.

The Attention program is made up of several transit phases, for as a rule operator intervention is of an intermittent nature.

Servicing the Timer. The equipment of the YeS EVM computers includes the timer -- a clock -- which permits the system to keep time. By using the timer, the Supervisor traces the current time and controls the time interval. The timer calculates the time just like an ordinary clock. We shall consider that it calculates and indicates absolute time. The timer readings can be picked up and used by the operating system program and by the user program. For example, the Assignment Control program, using the timer readings, prints out a message about the time of beginning and end of the assignment and its execution time. Any program can pick up the current time of day independently of what division of the basic memory it is being executed in.

The Supervisor can use the timer as a clock which tells when a previously assigned time interval is up. The size of the time interval is determined by the problem program and it is used on inspection of it. The Supervisor grants the problem program two versions of using the time intervals. One of them is to establish the waiting time interval, that is, the time during which the program is not executed. Its execution will be continued on expiration of the required time interval beginning with the location where it was halted. Another version is to establish the operating time interval of the program, that is, the time during which it is allowed to be executed. On expiration of the given interval the normal execution of the program is interrupted, and the control will be transferred to its subroutine in which the previously planned operations with respect to processing this interrupt signal from the timer can be executed. For example, a subroutine can undertake special operations in the case where the planned limit for execution of the program has been exceeded. In the problem program it is impossible to use both versions of control of the time interval simultaneously.

There is only one timer system, and it can be used to control the time interval in only one program of those being executed simultaneously. It is attached to the division in which the program is executed which is using it for controlling a time interval. During this period the timer cannot be used in other divisions to control a time interval. The attachment of the timer to a division is realized by the operator during the functioning of the system. The operator knows the division in which a program is using the time interval control, and at the corresponding time attaches the timer to the given program using the TIMER command of the Attention program.

As a rule, the timer is attached to the division for the entire time the program is being executed in this division, using the time interval control. It is possible to attach the timer to one of the divisions also during generation of the system, and then all of the programs using the time interval control are executed in the division in series one after another.

When working with the timer it is necessary to consider that it reckons an absolute time interval. In the multiprogram mode this means that the time interval established by one program includes in its execution time not only this program itself, but also the rest of the programs simultaneously executed with it. For example, the program has established an operating time interval of 15 seconds. After 15 seconds its execution is interrupted even though during this time it has been executed, let us say, only 8 seconds, and programs in other divisions have been executed for 7 seconds.

The timer service subroutines are part of the Supervisor nucleus, and they are included in the nucleus depending on the generation parameters.

The timer, just as any other input-output device, must be included during generation of the system in the devices serviced by the DOS YeS system. If its presence is not indicated during generation of the Supervisor, it is considered that there is no timer on the computer, and none of the functions of working with the timer will be provided by the Supervisor. If the timer is included in the devices serviced by the system, then any program can use the timer to obtain the current time of day. The possibility of using the timer in the problem program to control the time interval must be requested specially during generation of the Supervisor. If it is not requested, then the supervisor will only provide for keeping the time of day, and the time interval control functions cannot be used.

Creation of Check Points. When writing the program specially if the execution time of the program is large, it is necessary to show concern in advance for its reliability and its stability with respect to possible equipment failures. The instability of the program with respect to equipment errors can lead to the fact that during execution of the computer the emergencies that arise will require starting its execution over again although a significant part of the work has been executed properly. All of the useful results will be lost, and it will be necessary to obtain them again if a plan has not been provided in advance for repeating the program from the check point.

The check point is a location in the problem program in which it is most convenient to have exhaustive information about the current state of the program and the system so that it is possible to continue the execution of the program from this point in case of emergency. An example of a place where it is recommended that a check point be created in the program having an iterative nature is the completion of one or several iteration steps.

The creation of a check point means storing and retaining the state (photographing) of the problem program at the required point in time. Repeatedly starting the program from the check point means to resume execution of the problem program from one of the created check points in the case where the execution of the given program is halted for any reason before normal completion. For example, one of the reasons causing halting of execution of the assignment can be failure of the magnetic tape mechanism used in the given program. On replacing the failed tape-drive mechanism and making use of the information stored at the time of creation of the check point, it is possible to restore the state of the program and the system and resume execution of the program at this check point.

In the DOS YeS, the state of the program and also certain information about the state of the system are stored at the check point. The check point information is stored in the external memory -- on magnetic tape or on a disc in accordance with what the user has instructed the unit.

The indication of the location at which a check point must be set up is contained in the problem program itself. The programmer writes a special operator at that point in his program where the check point must be created. The direct storage of information (the creation of the check point) is done by the Supervisor. At the check point it stores, for example, the information such as the contents of the memory division in which the problem program is executed, the contents of the common registers, information about the segments of the files on the discs used by the program and the information about the position of the magnetic tapes. However, this information is still insufficient for organization of the check point at any location of the problem programs and starting from it. The check point does not contain, for example, information about the auxiliary functions of the Supervisor used in the problem programs (the presence of its own subroutines for processing the program failures and organization of communications with the operator, the timer readings); it does not contain information about the position of the data carriers on all of the other input-output devices beside magnetic tape, and so on. All of this information must be restored when starting at the check point. Thus, the location in the problem program where the check point is created must be carefully selected by the programmer.

The possibility of creating check points is provided for by the Supervisor always independently of the generation parameters. The programs for creating the check points are Supervisor transits.

Completion Program. Information must necessarily go to the Supervisor about the beginning and end of execution of the problem program. On receiving a request to execute the program, the Supervisor readies the system and in turn, its operating tables for activating the program. The Supervisor receives an indication that it is necessary to start up a program from the Assignment Control, it starts up the program, it monitors its execution, processes all of its requests, controls the errors that appear, and so on, but it does not establish the rules determining the operating time of this program. The completion time of it is determined by the problem program itself, and this is communicated to the Supervisor. The mission of the Supervisor is on completion of the program to remove the information about the completed program from its work tables and inform the operator of this, sending a completion message.

After completion of the program, the division in which it was executed is considered free; all of the peripheral devices which were used by the completing program are also free, and the entire division is ready to accept a new program. In order to accept the next assignment from the input flow of the free division and to prepare the system for its execution, the Supervisor calls the Assignment Control program into the free division. Thus, after completion of one program the system automatically proceeds with the acceptance and execution of the next program without operator intervention. This program completion mechanism permits package processing of assignments.

The reasons for completion of the program can be the following: the natural end of execution, the halting of execution plan in the program itself on occurrence of defined conditions; the request of the operator to halt the execution of the program; any failures (the program failures, input-output failures) making further execution of the program impossible. Only the natural end of execution is normal conclusion of the program; all the remaining causes, independently of the source of their occurrence, are perceived by the system as abnormal conclusion. On recognition of the situation indicating an abnormal conclusion of a problem program, the system halts the execution of the assignment (dumps the assignment), reports this to the operator, indicating the cause of abnormal conclusions and executes certain additional operations (for example, dumping the division of the basic memory in which the program was executed).

The basic memory is dumped if the problem program has requested a DUMP in its indication of an abnormal conclusion and also for any abnormal conclusion if the DUMP regime has been set up in the Supervisor. The DUMP mode can be established by the programmer in his assignment or it can be established for all assignments, and then it must be indicated as a standard mode of the Supervisor during generation.

Just as in the case of normal completion of a program, the Supervisor, on execution of all of the procedures with respect to abnormal conclusions, calls the Assignment Control program for acceptance of the next assignment.

This means that independently of normal or abnormal conclusion of the assignment, package processing can be used.

Use of the Supervisor Functions in the Problem Program. The Supervisor executes the majority of its functions by problem program request. These are, for example, the execution of the input-output operations, loading the program phase in basic memory, creation of a check point, and use of the timer.

Every request to the Supervisor is a defined sequence of instructions contained in the problem program. These instructions prepare the information which must be transmitted to the Supervisor, and they realize reference to the Supervisor directly with the help of a special instruction which is available in the YeS EVM computers. In order to facilitate the coding of these requests for the programmers writing their program in Assembler, a set of special operators are provided -- the Supervisor communication macroinstructions. The macroinstruction operation code defines the function which must be executed by the Supervisor, and the parameters give the information required by the Supervisor to execute the function. During the translation of the initial problem program using macroinstructions, each such macroinstruction is replaced by a series of the Assembler operators -- macroexpansion selected from the corresponding macrodefinition as a function of the parameters given in the macroinstruction. The macrodefinitions of all of the macroinstructions for communication with the Supervisor are stored in the library of initial modules. The flow chart for replacement of the macroinstruction by the corresponding macroexpansion is given in Fig 9.

In the DOS YeS system there are macroinstructions for requesting the following Supervisor functions in the problem program:

Loading the program phases into basic memory;

Starting the input-output operations and waiting for completion of them;

The creation of check points;

Indication of the presence of the programs on subroutines for processing program failures and for organizing communication with the operator;

The use of the timer to obtain the current time of day or control the time interval;

Indication of normal or abnormal completion of the programs;

Indication of the necessity to print out a defined segment of the basic memory or an entire division in which the program is executed;

Obtaining information from the communication buffer.

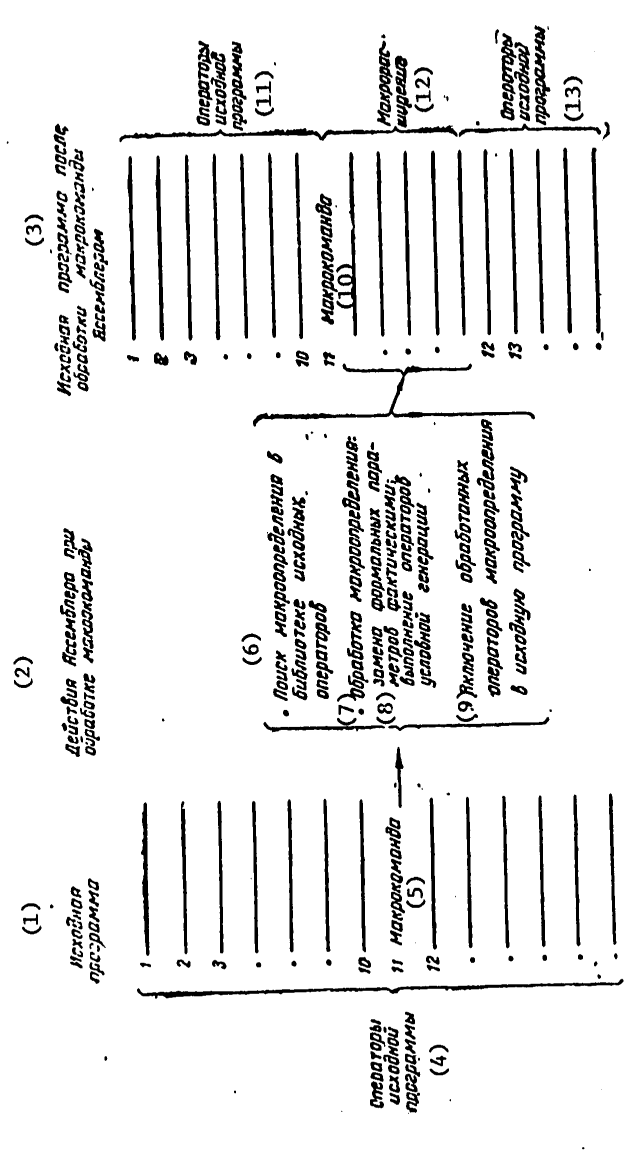

Figure 9. Replacement of the macroinstruction by macroexpansion

Key:
1. Initial program
2. Operations of the Assembler during processing of the macroinstruction
3. Initial program after processing the macroinstruction by the Assembler
4. Operators of initial program
5. Macroinstruction
6. Search for macrodefinition in the library of initial operators
7. Processing the macrodefinition:
8. Replacement of formal parameters by actual parameters; performance of the provisional generation operators
9. Inclusion of the processed macrodefinition operators in the initial program
10. Macroinstruction
11. Operators of initial program
12. Macroexpansion
13. Operators of the initial program

All of the functions except the last were characterized previously when describing the properties of the Supervisor. Let us explain what the macroinstructions are used for by means of which the problem program can receive information from the communications region or common.

The common is a segment of the basic memory in the controlling program or monitor zone containing information such as the name of the assignment, the program switches, the address of the beginning and end of the division, the address of the end of the program, and the calendar date. The problem programs executed in different steps of the same assignment can transmit information to each other through the common. The common zone contains a great deal of other information which is used in the operation of the operating system itself by the controlling program or monitor and the other programs of the DOS YeS system. Information is put into the common for the most part by the Supervisor. During operation of the system, a defined part of the common can be used by the problem program. For example, the problem program can set and read the program switches, obtain the calendar date and the name of the assignment in order to include them in its output summary or reports. The problem program can obtain information from common or change the contents of common by using macroinstructions.

The use of a defined group of macroinstruction presupposes that the Supervisor knows how to realize the simplified functions. Many of the Supervisor functions are generation parameters, including the functions which can be simplified by the problem program.

The basic Supervisor functions are listed in Table 2, and the ones of them which are generation parameters are noted. By the properties executed by the Supervisor on request from the problem program macroinstructions are presented by means of which these properties are requested by the problem program. For example, the creation of the check point is requested by the problem program by using the macroinstruction CHKPT.

Table 2

Basic Functions of the DOS YeS Supervisor

Function	Generation parameters	Macroinstruction for communication with Supervisor
(1)	(2)	(3)
Multiprogramming	Yes	-
Memory protection	Yes	-
Program (phase) loading	No	FETCH, LOAD
Organization of communications with operator	No	-
Starting the input-output operations	No	EXCP, WAIT, CCB
Processing the input-output interrupts	No	-
Standard processing of input-output failures	No	-

[Table 2, continued]

(1)	(2)	(3)
Keeping statistics on the magnetic tape failures	Yes	-
Standard processing of program failures	No	-
Processing of machine failures	No	-
Completion of assignments	No	EOJ, CANCEL
Dumping memory	Yes	DUMP, PDUMP
Servicing the common area	No	COMRG, MVCOM
Creation of a check point	No	CHKPT, STXIT, EXIT
Use of user subroutines in the problem program:		
For communication with the operator;	Yes	
For processing program failures	Yes	
Protection of the disc files	Yes	-
Isolation of the feed instruction from the channel program [driver] for the discs	Yes	-
Use of discs for system logic	Yes	-
Bufferization of messages to the operator	Yes	-
Servicing decimal and floating-point arithmetic	Yes	-
Servicing the timer	Yes	SETIME, GETIMB
Control of a time interval	Yes	TECB, WAIT
Distribution of basic memory among divisions	Yes	-
Adjustment for the required composition of the input-output equipment	Yes	-
Determination of the number of logical units for each of the divisions	Yes	-
Establishment of standard assignments for the logical units	Yes	-
Establishment of the standard operating conditions of the system	Yes	-

Assignment Control

The basic purpose of the Assignment Control program is to accept the assignment from the input flow of assignments and prepare the system for execution of the received assignment. In other words, the Assignment Control program must receive the information-description of the operation, the execution of which is required, and this information-description is placed in the operating tables of the controlling program for use of it during the entire assignment execution time. The description of the operation includes the information such as the program name, the names of the files with which the program will work, the correspondence between physical and logical units used in the program, specific operating conditions of the system required for performing

the planned operation. Without having all of this information, the system cannot perform the operations.

The description of the operation is put together by the programmer in the Assignment Control language. The writer is provided with the descriptions by the controlling operators permitting standard preparation of all of the information required by the system for execution of the operation. An example of the input flow of assignments is presented in Fig 10.

The assignment is received and processed by the Assignment Control program. It picks up all of the controlling operators from the initial division flow, controls their correctness and prepares the system for execution of the received assignment.

The Assignment Control program functions in the system at the time of transition from the execution of one program to the execution of another.

Any time when the execution of a problem program is completed, the control is transferred to the Supervisor which calls the Assignment Control program into the basic memory. Without requiring any additional instructions from the operator, the Assignment Control receives the next assignment from the input flow of assignments. The Assignment Control reads and processes the controlling assignment operators one after the other until the operator is read which requests execution of the problem program. On receiving this operator, the Assignment Control considers that the description of the operation is complete and the assignment contains no other information required for execution of the requested program; consequently, it is possible to send the Supervisor a request to load and execute the program.

The requested program is loaded into the basic memory and executed. After the execution of the started program is completed, the Assignment Control will again be called by the Supervisor into the basic memory for acceptance of the next assignment operators. Among the remaining operators there can be operators which describe other assignment steps. A special operator -- the end of assignment operator -- informs that there are no more steps in this assignment. If it is discovered that all of the steps (or the only step) of the assignment have been executed, the Assignment Control program initializes the system, removing all information from it pertaining to the completed assignment and proceeding with the acceptance of the next assignment.

The described flow chart presupposes that the initial flow of assignments is constant. If for any reason the input flow is exhausted (the assignments have been completed or the operator has not put them in the initial assignment flow), the Assignment Control program reminds the operator of the necessity for preparing the SYSRDR unit. The package processing in the division where the input flow has ended is halted. The Assignment Control waits for the operator's decision. The operator can prepare a new input flow, putting the assignments in the SYSRDR reader, report that the device is ready and after this the package processing in the divisions will resume.

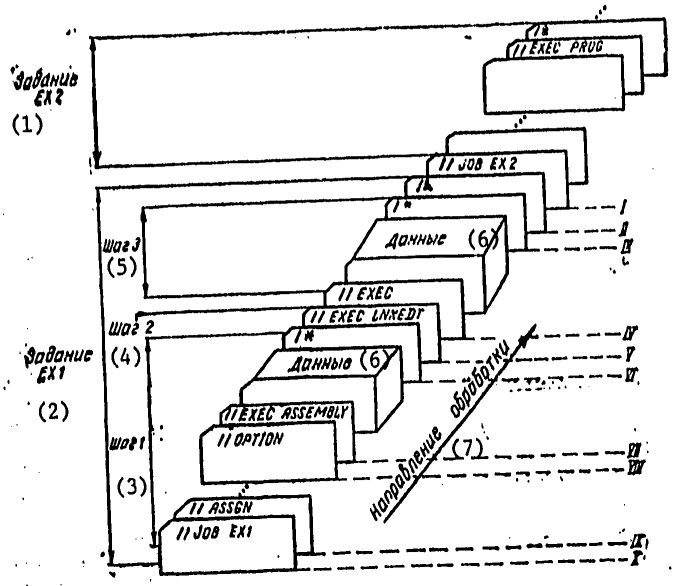

Figure 10. Example of the input flow of assignments: I -- beginning of the EX2 assignment; II -- end of the EX1 assignment; III -- end of the program data and step 2; IV -- execution of the edited program; V -- call of the editor for editing the program; VI -- end of the Assembler and step 1 data; VII -- Assembler call operator; VIII -- operator for establishment of the required system modes; IX -- operator for assignment of a physical device as logic; X -- beginning of the EX1 assignment.

Key:
1. EX2 assignment
2. EX1 assignment
3. Step 1
4. Step 2
5. Step 3
6. Data
7. Direction of processing

A characteristic feature of the operation of the Assignment Control program is the fact that it always is executed in the problem program zone in contrast to the Supervisor which is always located and executed in the monitor zone. Here the Assignment Control is called by the Supervisor into the basic memory division in which the execution of the program has been completed. Thus, the Assignment Control receives the assignment in the division in which the requested program will be executed. After output of the request to the Supervisor to load the problem program the Assignment Control is not needed in the division, for at this time all of its functions with respect to preparation for execution of the problem program

have already been executed by the Assignment Control. Therefore the Supervisor considers that the entire division is free and the required problem program can be loaded in it.

All phases of the Assignment Control program are permanently stored in the library of absolute modules, and from it the Supervisor again calls the Assignment Control into the basic memory any time the execution of a program of the DOS YeS system itself or a program prepared by the user has completed its execution.

On receiving information from the Assignment Controlling operators, the Assignment Control program uses this information to execute the following basic functions:

Designation of physical devices as logical units;

Establishment of the operating mode of the system;

Editing and storage of information about the volume and file marks;

Preparation of the program for repeated starting from the check point.

The information required by the Assignment Control for performance of the assignment functions can be delivered not only by the programmer, but also the operator. In order to delimit possibilities which the Assignment Control grants to the programmer and the operator, there are control operators and directives in the Assignment Control language: the programmer uses the controlling operators to compile the operating assignment, and the directives are used by the operator, who enters them in the input flow of assignments or from the panel typewriter. The list of controlling operators and assignment control directives is presented in Table 3.

Assignment of Physical Devices as Logical Units. As has been noted, by using the apparatus of logical units the DOS YeS system insures independence of the programs with respect to the specific physical devices. This apparatus contains the procedures for designating the physical devices as logical units and also a mechanism for maintaining and using the established correspondence during the operating process.

Many devices of one type can be connected to the YeS EVM computer. This creates defined conveniences in operation and increases the system reliability. For example, if a device has failed, or is in preventive maintenance, the system can operate with another device of the same type. The YeS EVM computers which are available to the different users can have a different number of devices, and the physical addresses of these devices are also different. The desire to make the programs independent of the specific address of the input-output device is natural. The program which is adjusted to the address of the specific device directly before execution itself can be executed on any of the computers which include the types of devices used in this program. It is convenient for the users to exchange such programs, for they can be executed in a required configuration of the devices without any alterations.

Table 3

List of Controlling Operators and Directives of the Assignment Control

Purpose	Controlling operator	Directive	Function
Assignment identification	//JOB		First assignment operator determines the name of the assignment
	/&		Last assignment operator
File definition	//TLBL		Reports information needed for the creation and checking of the file marks on the magnetic tape
	//DLBL		Reports information needed for the creation and checking of the file marks on the disc
	//EXTENT		Reports information on the section protected by the file on the disc
	/*		End of file (assignment step)
Transmission of information to the program	//LBLTYP		Determines the basic memory size which must be reserved in the problem program field for processing the tags
	//OPTION		Establishes the operating condition of the system
	//UPSI		Establishes the program switches in the common area
	//DATE		Establishes the date for the program
Control of the flow assignments	//PAUSE	PAUSE	Halts the input of the controlling operator; the system waits for the operator's instructions
		CANCEL	Assignment removed
		STOP	Package processing in the division halted temporarily, and this division is in the waiting condition until the operator starts it again using the Attention instructions
		UNBATCH	Package processing cancelled in the front plan division, and the division is cleared

[Table 3, continued]

Purpose	Controlling operator	Directive	Function
Establishment of system parameters		ALLOC	Permits the operator to redistribute the basic memory among the divisions
		MAP	Causes the distribution table for the basic memory among the divisions to be printed out on the panel typewriter
		SET	Is used to establish the date, time of day, and program switches in common, the number of rows on a page of systems text and certain other information
Communication with operator			Permits placement of any operator comment in the assignment
	//LOG	LOG	Indicates the necessity for system printout or output to the panel typewriter of all of the controlling operators and assignment directives
	//NOLOG	NOLOG	Cancels the LOG regime established by the operator or directive
		KT	End of communications with the operator
Control of the input-output devices	//ASSGN	ASSGN	Used for designation of the specific input-output devices a logic unit
	//RESET	RESET	Establishes standard or permanent designations for logical units
	//LISTIO	LISTIO	Used for printing out the table of assignments to the logical units
	//CLOSE	CLOSE	Used for completion of work with the files on magnetic tape or on discs
	//MTC	MTC	Indicates the control operations for the magnetic tape
		DVCDN	Permits the operator to communicate to the system that the specific input-output device has become unfit and cannot be used by the system

[Table 3, continued]

Purpose	Controlling operator	Directive	Function
		DVCUP	Permits the operator to communicate information to the system that the specific input-output device forbidden by him with respect to the DVCDN directive can be used by the system
		UNFA	Permits the operator directly to cancel the designations of all of the logical units in the front plan division
		UCS	Establishes correspondence between the binary codes for the information to be printed out and the printed symbols
Execution of the program	//EXEC		Requests execution of the program, the name of which is indicated in the operator
	//RSTRT		Realizes starting of the program from the check point

The independence of the problem programs with respect to the addresses of the input-output devices is insured by the fact that the programmer is permitted to indicate only the symbolic name and type of this device for each unit used, that is, it is not the address of the input-output device that is indicated, but some "logical unit" defined by the symbolic name. Thus, specific addresses of the input-output devices are excluded from the program, and the program deals only with the logical units. Obviously in order to execute the program in which the logical units are used, directly before execution the logical units must be placed in correspondence to the addresses of the actual units available on the computer. The procedure for establishing correspondence of the actual and logical units is called the assignment of the physical unit of the logical unit.

The information about assignment of the physical units of logical goes to the programmer or the operator in the assignment to work with the help of the controlling operators or the directives of the Assignment Control language. The Assignment Control program realizes the designation requested in the assignment. The designation established by it is used by the system when executing all of the input-output procedures.

The symbolic means which the programmer can use in his program for referencing the devices are written in the form SYSnnn where nnn is the number of the interval from 0 to 244 or special values of RDR, IPT, PCH, LST, LOG, LNK, RES, SLB, RLB can be used as nnn. The SYS000-SYS244 logical units

are called the programmer logical units; specific devices of any type can be designated for them. For example, the SYS001 logical unit in one program can be used for operation with the unit for input from punch cards, and in the other for working with magnetic tape.

The logical units of the SYSRDR, SYSIPT, and SYSLST type are designated system logical units, for they are used by the programs of the DOS YeS system itself and admissible types of devices are established for them. However, this does not mean that the system logical unit cannot be used in the problem program. When necessary the problem program can make use of these logical units for the same purposes as the DOS YeS system itself.

In Table 4 a list of system logical units is presented, and the types of physical devices which can be assigned to them are indicated.

The Assignment Control sees to the correctness of the assignments indicated by the programmer and the operator, for example:

The same physical unit cannot simultaneously be assigned to different divisions (except the designation for SYSRES and SYSLOG);

In certain cases and in one division one and the same physical unit must not be designated for the various logical units;

For the system logical unit, the physical device must be designated among the admissible types of physical devices for it.

Table 4

List of System Logical Devices

System Logical Devices		
Name	Purpose	Type of physical device
SYSRES	System residence	Disc
SYSRDR	System assignment input	Punch card input Magnetic tape Disc
SYSIPT	System input	Punch card input Magnetic tape Disc
SYSPCH	System punch	Punch card output Magnetic tape Disc
SYSLST	System print	Printer Magnetic tape Disc
SYSLOG	System register	Panel typewriter Printer
SYSLNK	Data input for Editor	Disc
SYSRLB	Personal library of the target modules	Disc
SYSSLB	Personal library of the initial modules	Disc

The system distinguishes three types of purposes: standard, permanent and temporary.

The standard is the purpose which is carried out during the system generation. It is characterized by the fact that the adopted correspondence of the logical and physical devices goes into force from the time of execution of the initial loading procedure, and it remains during the entire time of operation of the system. The standard designations are made only for the background division, and they are absent for the front plan divisions.

The purpose which the operator carries out with the help of the designation directives can be permanent or temporary. The permanent designation cancels the standard designation for the logical unit and then for the entire time that the system operates it acts as a standard. The permanent designation can be cancelled either by a new permanent designation or by the initial loading procedure. At each specific time the standard and permanent designations are equivalent with respect to use by the system. They are distinguished by the fact that the standard designation is established during generation of the system, and the permanent one, during work with this system. Standard designation is restored during the initial loading procedure.

The temporary designation is characterized by the fact that it remains active only within the limits of one assignment and replaces the standard or permanent designation during this time. After completion of the performance of the assignment, all the temporary designations for the logical devices are cancelled, and the standard or permanent designations for them go into effect.

Using the control operators of the Assignment Control, only the temporary designations of logical addresses of the devices for the logical units are used, that is, all the designations which the programmer makes remain valid only within the limits of its assignment. For the next assignment its own designation must be made. In the cases where the standard or permanent designations coincide with those which the programmer requires, it is unnecessary to prepare the controlling designation operators for the assignment. The system time and the programmer's ethics are saved.

For example, for the SYSRDR logical unit during the system generation, a specific card input unit can be designated. The programmer can change his purpose, designating a specific magnetic tape in his assignment for the given logical unit. After completion of the assignment, the standard designation is restored, that is, for the SYSRDR logical unit the card reader is designated.

At the same time, just as with the help of the controlling operators of the Assignment Control it is possible to perform only temporary assignments, the permanent assignments of the operator remain valid from assignment to assignment. This offers the operator the possibility of operative intervention in the computing process in cases where it is necessary for any reason to replace one physical unit by another.

The role of the operator is especially great when assigning the units for the front plan division. For these divisions the standard designations are made during generation of the system. All of the designations for the front plan programs, as a rule, operate only within the limits of one assignment. Only the operator can indicate to the system that in the given front plan division the designations made for a specific assignment must be retained even after completion of the assignment. Thus, these designations become permanent, and their action is analogous to the action of the permanent designations for the background division. They remain valid until the operator tells the system to cancel these designations or on execution of the initial loading procedure.

All of the designations of units are realized by the Assignment Control program. After completion of the assignment it cancels all of the temporary designations and restores the permanent and standard designations which the operator has not changed during the system operation.

Establishment of System Operating Conditions. The program components of the DOS YeS system provide the programmer with a number of services, each of which can be used independently of the rest. Thus, the translator from the Assembler language can if the programmer desires punch objective modules or not do this; the Assignment Control can print out all of the controlling operators on the typewriter or not. The DOS YeS system has means to indicate the required type of services in each assignment. The set of indicated services creates a defined use condition for the system or operating condition of the system. It is possible to consider the conditions established for the operation of the translators and the Editor, for the Assignment Control program itself, the program completion conditions and also the operating conditions of the system with information about the file marks separately.

Each translator provides the capabilities which it executes or does not execute depending on whether the programmer requests them or not. For example, depending on the programmer's instructions the translator can:

Place the objective module on punch cards, magnetic tape or disc;

Put the objective module on a disc for subsequent editing;

Print out the initial module;

Print out the list of all errors detected in the initial program;

Print out the table of symbolic names used in the initial program.

This type of information is output by any translator, and therefore it is important to standardize the form in which these instructions are given to the translators by the programmer and also to determine the times when the programmer must give these instructions.

In the DOS YeS system, the instructions to the translators about their operating mode and also determination of any other operating regime of the system are given by the programmer in the assignment describing its operation. The Assignment Control receives the programmer's instructions, determines the requested modes and sets them up for mandatory execution by the system programs which pertain to these modes. For example, the purpose of the assignment can be the performance of such operations as the translation of one initial module written in Assembler, translation of another initial module in Fortran, combination and editing of them to obtain one program ready for execution and the execution of the program obtained directly. Here, the initial program must be printed out, lists of symbolic means used in the program obtained, and in the case of abnormal completion of the prepared program, a photograph of the division must be obtained. This is the statement of the problem. For realization of it, in addition to the text of the initial modules in Fortran the programmer must prepare the assignment in which by using the Assignment Control operators he indicates his requirements to the system:

Print out the initial module;

Print out the list of symbolic means;

Indication to the translators to write the target modules obtained as a result of translation on the disc;

Set up the DUMP mode for the time of execution of the prepared program.

The first three instructions will be realized by the Assembler and Fortran translators, and the fourth, by the Supervisor.

The operating mode of the system is established as the operating mode only for the components of the operating system of the DOS YeS and especially for the translators. Since all of them can be performed only in the background division, it is meaningful to talk about the operating mode of the system only for the background division programs. The operating mode of the system can be given on generation of the system (standard mode), or it is established by request of the programmer for his assignment. In the specific system the mode which permits the execution of the following, for example, can be standard:

Print out of the Assignment Control program operators on the panel typewriter and these operators are input to the computer;

Print out of the contents of the basic memory in the case of abnormal completion of the assignment;

Punching of the card columns with objective modules after translation;

Print out of the initial modules.

The standard operating mode is maintained by the Assignment Control program. Each specific module can be altered by the programmer by using the controlling operators. The mode established by the programmer operates within the limits of the assignment. After completion of execution of the assignment the Assignment Control program restores the standard mode. For example, let the standard mode be adopted in which the translators must punch the target programs. The programmer can cancel the punching of the cards for his assignment. After completion of the assignment the standard mode providing for punching will be restored by the Assignment Control program.

Editing and Storage of Information About the Volume and File Marks. The Assignment Control program executes a number of procedures with respect to preparing information on the volume and file marks.

In the DOS YeS system the processing of the files which are on magnetic tape and discs includes the execution of defined operations connected with checking whether the file belongs to the problem program. In order to avoid the situation where the file belongs to the problem program, the DOS YeS establishes the set of rules for file organization and methods of working with it. Among these rules is the requirement that the files on magnetic tape and disc have marks of completely defined structure -- system marks; in order that operation of the file begin with the execution of the opening procedure -- the procedure which, by using the file and volume marks, establishes correspondence between the file and the problem program processing this file. The correspondence is considered established if the mark written on the carrier together with the file coincides with the information about the file which the programmer indicated in his assignment. The information must be indicated in the assignment about the used files which the system needs in order to be convinced of the correctness of preparation of the carriers with the files and to permit the problem program to work with the requested files.

The information about the files (the volume and file marks) is supplied by the programmer using the controlling operator. The Assignment Control program receives these controlling operators, edits and stores the information contained in them in the system residence for future use of it by the input-output control system programs.

In the system residence a special section the size of one cylinder is isolated in which the information about the marks is stored. It is called the mark cylinder and is separated into four regions. The first three regions are set aside for storing information about the marks coming from each division of the memory. The first two tracks of the marks are allocated to the background division; the next two are allocated to the second division of the front plan; and then two are allocated to the first division of the front plan. The next four tracks of the mark cylinder form the fourth region designed for storing information about the file marks with which the programs of any division of the basic memory can operate. As a

rule, in this region information is stored on the file marks which is used by the system service programs and the translators of the DOS YeS system. These programs operate with the files which are called system files, and the region of the mark cylinder which is designed for storing information about the marks of the system files is called the mark region of the system files or the mark region of the system.

The information is entered in the mark region of the system files usually during generation of the system, and it is permanently retained during operation. This relieves the programmer of the necessity for preparing controlling operators for a specific assignment describing the marks of the system files. At the same time the programmer has the possibility of putting information about the marks of the files needed by him in this region. This pertains both to the information about the marks of the system files and about the marks of any other files with which the problem program works.

The Assignment Control program stores information about the marks delivered using the controlling operators always in one of the mark cylinder regions: the mark region of the system files or in the mark region for the same region in which the assignment is accepted. Depending on whether the information about tne marks is to be used by the system programs, the programs of one assignment or several assignments executed in the same division, the programmer indicates in which of the regions of the mark cylinder it must be entered. He gives this instruction using the controlling operators establishing the required operating mode with information about the marks.

The programmer can require that all the information about the marks delivered by his assignment be written in the mark region of the system files. The information written in this region about the marks can be used by all of the programs executed in any division, and it is kept in the mark cylinder until the programmer requires that new information about the marks be entered in this region.

In the case where the information about the marks is designed for use in the programs executed in the same division, it is written in the mark region of the division. Here there is the possibility of indicating which of the two tracks of the mark region of the division it must be entered in. Information about the marks is written on the second track. It can be used in different assignments performed in this division. The information written on this track is stored from assignment to assignment until the request to enter new information for this division comes in the next assignment. Information about the marks which is used only in one assignment and is retained only for the time of execution of the assignment is entered in the first track of the division mark region. The placement of information about the file marks on the mark cylinder is illustrated in Fig 11.

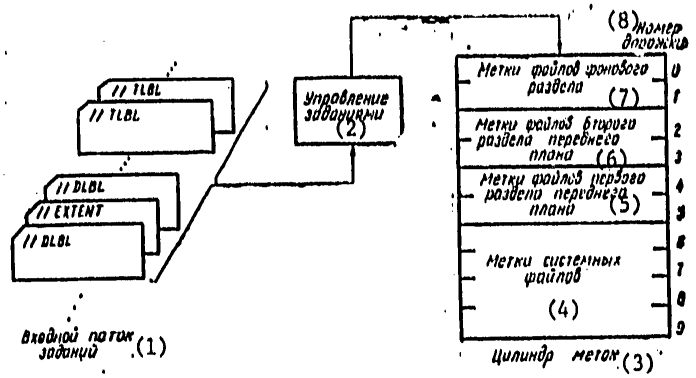

Figure 11. Placement of information on the mark cylinder

Key:

1. Input flow of assignments
2. Control of assignments
3. Mark cylinder
4. System file marks
5. File marks of the first division of the front plan
6. File marks of the second division of the front plan
7. File marksz of the background division
8. Track number

Repeated Starting of the Program from the Check Point. The Assignment Control program offers the possibility of repeated starting of the program from any check point which has been created during the program execution. The following basic requirements must be observed in this case. The repeated starting of the program from the check point must be realized in the same division in which the program was executed at the time of creation of the check point. For all the logical points used in the program, the same or other physical input-output unit must be designated.

The input-output files used by the program must be put into the condition corresponding to their condition at the time of creation of the check point, for the program for creation of the check point does not provide the means for setting up the input-output files for the required recording on such snits as the input-output punch cards, the input-output punch tapes and the printers. The programmer himself must consider these means and communicate them to the operator when this is needed. It is also necessary to remember that in the case of repeated starting from the check point the problem program must again communicate all information to the Supervisor about the presence in it of its own subroutines for processing the program failures and for organization of communications with the operator and information about the use of the timer.

Initial Loading

The functioning of the operating system of the DOS YeS system begins with the execution of the initial loading procedure. The permanent goal of this procedure is to ready the computer and the DOS YeS system which will function with it for operation.

The initial loading procedure begins with the execution of the ready operations by the operator in the following order:

Installation of the set of discs which contain the residence file on the selected unit;

Setting up the address of this disc in the load switches located on the operator panel;

Pressing the "load" button.

After the "load" button is pressed, the permanent program is automatically entered into the basic memory from the system residence. This program, after receiving control, loads the Initial Load program into the basic memory.

One of the basic functions of the Initial Load program is to read the Supervisor nucleus into the basic memory. After the Supervisor nucleus is loaded, other functions of the Initial Load program can be executed among which the main one is indication to the DOS YeS system of the specific configuration of the input-output devices of the given computer among the types of devices, the servicing of which has been requested during generation of the system.

During generation of the DOS YeS system for the specific computer, the types and the number of input-output units were indicated which can be connected to the computer. Specific physical addresses of these units can also be indicated. The specific DOS YeS system obtained as a result of generation has been used on the computer for a prolonged time interval, for the generation procedure is quite tedious and takes up a great deal of time, and the reasons which would require a new version of the operating system are not frequently manifested. If the set of input-output devices is altered on the computer, this is not reason for generation of a DOS YeS system, for these applications can be considered in the majority of cases when executing the Initial Load program. The changes in the set of units by comparison with the set which was indicated during generation (changes in the physical address of a unit, failure of a unit, addition of a new unit of the admissible type, which does not cause the total number of units given during generation to be seated) can be indicated in the system when executing the Initial Load procedure.

If the set of physical input-output devices has not been specified during generation of the system, it must be indicated when performing the Initial Load procedure. Thus, the physical unit can be connected to the DOS YeS system either during generation of the system or during execution of the Initial Load procedure. If the unit is not connected to the DOS YeS system, then it is impossible to work with it in the future: the DOS YeS system considers that it is not available to the computer.

The instructions pertaining to additions of new units to the system or elimination of those which will not be used for a defined, long time interval, are provided by the operator by using directives. These directives can be received by the Initial Load program from the panel typewriter or from the card input reader. Thus, the operator has the possibility of operatively changing the composition of the input-output units with which the system will operate in the future.

All of the changes in the set of units which the operator has expressed in the Initial Load are entered only in the corresponding tables of the Supervisor and the basic memory; they are not entered in the residence file, and therefore they are valid only until a new Initial Load procedure is executed.

After the changes in the configuration are indicated to the system and also when no alterations need to be made, the operator must indicate the date, and if a timer is present in the system, the time of day.

On receiving the date from the operator and, if necessary, time of day, the Initial Load program completes its work by transferring control to the Supervisor. It, in turn, calls the Assignment Control program into the basic memory which is adjusted to receive the first assignment in the background division.

After execution of the Initial Load procedure the Assignment Control must realize permanent designations for the system logical units which have not received standard designations during generation. Frequently, in order to emphasize the time when these designations are made, it is said that the designations are made when executing the Initial Load procedure.

CHAPTER 3. DATA CONTROL

The programming of any input-output operation is a tedious problem and requires great effort on the part of the programmer. In the opinion of the specialists, the tediousness of the writing and debugging of the input-output operations when programming the majority of economic problems is estimated at 40% of all of the programmer expenditures. At the same time the set of data processing programs on like devices (punch card units, magnetic tapes, discs) have much in common. The work of the programmer can be facilitated if he is granted the standard data control means for all input-output devices of the YeS EVM system. The most complex operations requiring special care with respect to input-output programming will be picked up from the programmer if the data control assumes the execution of such procedures as the delivery of data from the external carriers to the basic memory (data input) and, on the contrary, the transfer of data from the basic memory to the external carrier (data output). The basic purpose of the data control also consists in the performance of these two important missions: data input and output.

In the DOS YeS system the data control functions are performed by the input-output control system. In the system there are two data control levels: logical and physical. Each control level is realized by a set of programs called the physical input-output control system and the logical input-output control system, respectively.

Physical Input-Output Control System

The physical input-output control system is a component part of the Supervisor. It is used by all the programs of the DOS YeS system independently of whether they use the logical input-output control system or not. The logical input-output control system also is based on the physical system.

The physical input-output control system executes the functions with respect to organization of work with the input-output devices, such as the keeping of the request queues for the performance of the input-output operations, the startup of the input-output operations, the processing of all of the input-output interrupts, the processing of the input-output errors. Each

of these functions was discussed previously when describing the Supervisor properties. The enumeration of these functions alone indicates that the physical system relieves the programmer of the execution of a significant part of the work connected with the planning and organization of the operation of the input-output units. At the same time the physical system charges the programmer with all responsibility with respect to the description of the input-output operations themselves. The programmer must himself compile a channel program or driver for his input-output operations, and therefore he must know the operation of the input-output devices quite exactly.

Each time it is necessary to execute the driver, the programmer references the Supervisor. In this case he, in addition to the driver, sends the Supervisor a data control module. The purpose of this module is to transmit information in standardized form which is required for execution of the input-output operation from the problem program to the Supervisor and back, from the Supervisor to the problem program. In the data control unit the programmer indicates, for example, the name of the logical unit for which the driver must be executed; the address of the driver in the basic memory, and so on. After the input-output operation is executed, the Supervisor places the information about the completion of the input-output operation of interest to the programmer in the control unit.

The fact that even when using the physical input-output control system independence of the program with respect to the specific addresses of the input-output devices is insured, for the apparatus of the logical units operates on a physical level, is very important.

The programmer sends the Supervisor a request to execute the input-output operation using a special macroinstruction, a parameter of which is the address of the data control unit. The Supervisor processes it and returns control to the problem program without waiting for completion of the started input-output operation.

The problem program itself must decide whether it is executed or not without waiting for completion of the started input-output operation. If the execution of the problem program cannot be continued before completion of the input-output operation, the programmer must report this separately to the Supervisor, sending a macroinstruction to organize waiting. When the input-output operation requested by the programmer is completed and, consequently, the execution of the problem program can be continued, the Supervisor can transfer control to the problem program. The Supervisor places the information about the completion of the specific input-output operation of interest to the programmer in the data control unit. For example, the programmer can receive a message as to whether the operation has been completed normally, whether errors have been detected in the device and what these errors are, whether such special situations have arisen in the absence of punch cards in the hopper of the punch card reader, the end of a sheet of paper on the printer, and so on. The programmer must himself also provide for a reaction to all possible special situations in his program during the input-output programming on the physical level.

The separation of the duties among the programmer and the physical data control system takes place as follows. The programmer is obligated to compile and include in his program the drivers for each input-output operation. The physical data control system organizes the execution of the driver with respect to requesting the problem program.

A more developed level of input-output programming is the logical level realized by the logical input-output control system.

Logical Input-Output Control System

The logical input-output control system processes the data files. It deals with the logical recordings which are the basic units of information for the program processing the data. For the logical system the format of the recordings is significant as is the organization of the files, that is, the method of placing the recordings in the external carriers. The logical level guarantees the programmer standard procedures for the organization of the data input-output, at the same time relieving him of the necessity for knowing specific physical units. When using the logical control level, the programmer must be concerned only about the logical structure of his data. All of the functions connected with delivery of the data from the external carrier to the basic memory and back are assumed by the logical input-output control system. If for any reason the programmer does not eliminate the standard procedures on the logical level, he has the possibility of constructing his own procedures based on the physical level.

Files and Logical Recordings. In the data processing problems files are used which are made up of the most varied recordings which must be placed on the external carriers in such a way that they can be processed efficiently. The processing of the file includes reading the file recordings, processing them and storing the processed recording on the external carrier. The logical input-output control system takes on the execution of the reading and storage of the file.

In each specific case the file is characterized by the space it occupies on the external carrier and then the basic memory; the format of the recordings making it up; the principle by which the recordings in the file follow each other in sequence, and so on. In addition to other arguments, the solution of these problems is influenced by the selection of the type of peripheral device on which the file will be located. All of this together determines the organization of the file. In the DOS YeS system the files can have direct, series and index-series organization.

The file can contain entries of all three formats: fixed length, variable length and undefined length. The name of the format is determined by the length of the entries making the file up. The file is made up of fixed length entries if the length of all of the entries is identical; variable length if the length of the various file entries is different, but known for each entry; and the entries about the length of which nothing is known in advance have undefined length.

A significant requirement when forming the variable length entries is the necessity to pad fields made up of four bytes to each entry in which the length of file and the bytes is indicated.

For certain types of devices, entries of any of the indicated formats are permitted; for others, only certain formats are permitted. For example, the files on magnetic tape and discs can be made up of fixed or variable length entries or entries of undefined length; for the input punch card files, only fixed-length entries are permissible.

The file entries on the magnetic tapes or discs can be grouped into blocks to improve the efficiency. The entries combined into a block are called block entries. The blocking of the entry decreases the number of physical references to the external units and saves external memory, for the memory occupied by spaces is decreased.

The functions of blocking the entries can be performed by the logical system only for the entries of fixed and variable length. By instruction of the problem program the system itself groups the entries of fixed or variable length into the blocks with placement on magnetic tape or disc, and it also isolates the entries from the blocks and transfers them to the problem program for processing on reading from magnetic tape or disc. Since the system does not have information about the length of the entries of undefined length, it cannot execute the blocking functions for the entries of undefined length. The blocking of the entries of undefined length, if required, must be carried out by the programmer himself. The formats for the unblocked and blocked entries are presented in Fig 12.

Depending on whether the files are designed for input or output of the data, they are input or output data files. The input file is located on the input units, and the logical system sends the logical entries from the input unit to the basic memory. The output file is located on output units and the logical system creates it by moving the logical entries from the basic memory to the output unit.

The logical data control system gives special attention to the organization of the work with the files on magnetic tape and disc, for these devices are the carriers of the large files, and they are used most frequently for data storage. For the files on magnetic tapes and discs, the logical system executes them in addition to the functions with respect to data input-output directly, also such procedures as recognition of the required file, prevention of destruction of the file as a result of erroneous actions by the operator or program errors and also procedures increasing the reliability of the file processing. In order that the logical system be able to execute these additional functions, the files on the magnetic tapes and discs are equipped with marks. The logical system takes in itself the basic operation of the creation of file marks and the processing of them.

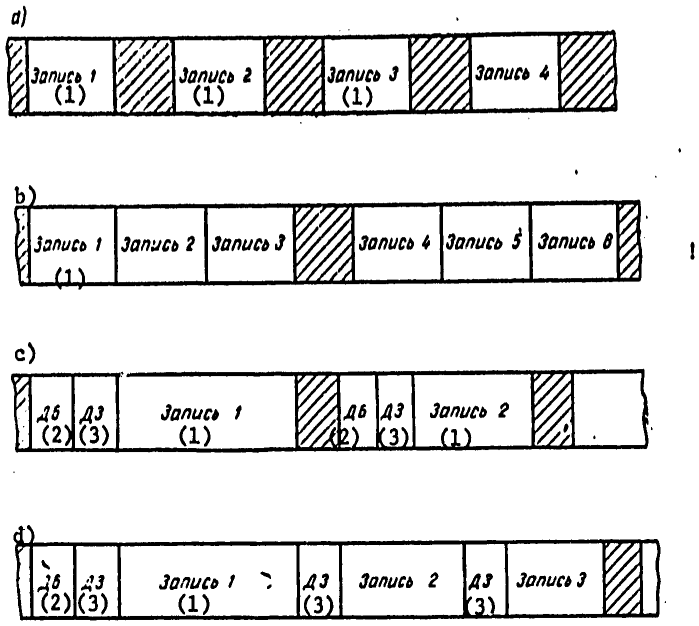

Figure 12. Formats for blocked and unblocked entries of fixed and variable length:
DB -- length of unit; DZ -- entry length.
a -- unblocked fixed-length entries; b -- blocked fixed-length entries; c -- unblocked variable-length entries; d -- blocked variable-length entries

Key:
1. Entry ...
2. DB
3. DZ

File Marks on Magnetic Tape. A characteristic feature of working with a magnetic tape by comparison with such carriers as punch cards and printed documents is the fact that the operator cannot himself be convinced that he actually is dealing with the required data file. When the work is done with punch cards or printed documents, the operator can determine by examination of the carriers themselves whether the required files are actually being processed. The situation is different when working with magnetic tape. The operator himself cannot read the information recorded on a magnetic tape, and he cannot be sure that he has installed the required magnetic tape. If there are a large number of magnetic tapes at the computer center, then in practice it is impossible completely to avoid operator errors connected with setting up the magnetic tapes. At the same time, the appearance of such errors is entirely inadmissible, for this is a disaster for the

data processing system. The standard procedure for avoiding such situations is the placement of special blocks called marks on the magnetic tapes.

In the DOS YeS system, the logical input-output control system can process magnetic tapes both with and without marks. However, the use of marks on magnetic tapes permits the programmer to avoid many unpleasantries when processing the data.

The marks on magnetic tapes can be standard and not standard. The standard marks have fixed structure, and they are divided with respect to purpose into system marks and user marks. The system marks contain defined information, they are created and checked by the programs of the logical input-output control system. Files having system marks can also contain user marks which must be processed by the problem program itself, and the logical input-output control system only reads them for processing or writes them. Thus, in the DOS YeS system the files on magnetic tape can have no marks or have only system marks or have system marks and user marks or have original nonstandard marks which the system did not participate in the processing of: the nonstandard marks must be processed by problem program itself. The standard marks on the magnetic tape are the following types: the volume marks, the initial file marks, the end of file marks, the end of volume of marks.

The volume mark is always written as the first block on the magnetic tape and is used to check the correctness of selection of the volume.

The initial file marks are entered in front of the file and are used by the input-output control system when searching the file on the magnetic tape. The initial system file mark will contain information about the files such as the file identifier by which the system recognizes the file; the registration number of the file establishing the communications between the volume and the file; the order number of the volume with respect to the volume in which the file begins; the order number of the file with respect to the other files with the same volume; the date of creation of the file and the state of expiration of the file storage time to insure storage of the file during the indicated time interval.

The end of file marks are written at the end of the file. Their structure coincides with the structure of the initial file marks, but they also contain information about the number of blocks and the logical entries in the file. This information is used by the system to check whether all of the entries in the file have been processed. For the files on magnetic tape this check is one of the forms of logical monitoring of the correctness of the processing. If the file occupies only one reel of magnetic tape (a multivolume file), then after the last block of the file on all volumes, except the last, end of volume marks are entered and not end of file marks. The marks are separated from the file itself by a special block

which begins with a tape mark. On the magnetic tape with standard marks, the tape mark always occurs after all of the initial file marks, and it precedes the data file; it exists also after the last block and precedes all the end of file or end of volume marks. Fig 13 shows the structure of the files on magnetic tape with standard marks.

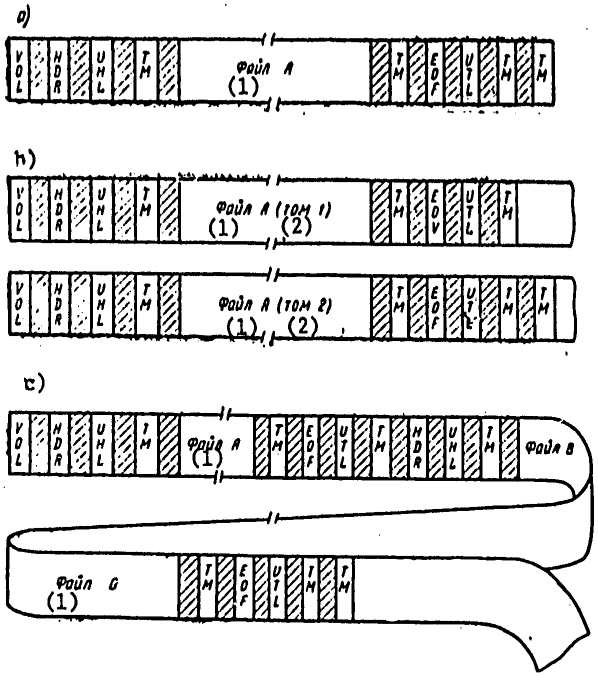

Figure 13. Structure of the file on magnetic tape:
VOL -- volume mark; HDR -- initial file mark; UHL -- initial file mark processed by the user (can be missing); TM -- tape mark; EOF -- end of file mark; EOV -- end of volume mark; UTL -- end of file mark processed by the user (can be missing), crosshatched sections -- gaps between zones (blocks) on magnetic tape.
a -- one file on one magnetic tape reel; b -- one file on several magnetic tape reels (multivolume file); c -- several files on magnetic tape reel (multifile volume).

Key:
1. file ...
2. volume ...

File Marks on Discs. The files on discs, just as the files on magnetic tape, must have marks. In the DOS YeS system the files on the discs must have standard marks independently of whether the physical or logical data controller was used to process them. The logical input-output control system cannot process the files on the discs which do not have standard marks. The standard marks are divided into system marks and user marks. The system marks are mandatory; they are processed by the logical input-output control system. In addition to the system marks, the files can have user marks which must be processed by the problem program, and the system only reads them from the discs or writes them.

In contrast to the magnetic tape, the system file marks on the discs are separated from the files themselves. The system marks of all files placed on one volume, that is, on one package of discs, are concentrated in one special file which also is on the same volume. This file is called the volume index. The user file marks on the discs, just as the file marks on magnetic tape, belong to the file itself and, as a rule, are located directly in front of the file.

The system file marks contain broad information about the file permitting the system to recognize the required file, determine the location of the file on the disc, establish the boundaries of the sections occupied by the file on the disc, and also obtain all of the information necessary to the system for processing the files on the discs. The system marks of all the files are in the volume index, and the information about the location of the volume index is contained in the volume mark. The volume mark is also written at a defined place on all of the disc packets. It is used by the system to recognize the volume and determine the location where the volume index is located. Fig 14 contains a schematic description of the placement of the files on the discs.

Data Acess Techniques. The DOS YeS system isolates several standard methods of organizing the data into files and the method of processing the organized data. The set of procedures for organizing the data and the method of processing the data form a defined access method for these data. In the DOS YeS system there are three access methods: series, direct and index-series.

The series access method presupposes that the data are organized in series, that is, the logical entries are located on the external carrier in the order in which they are then processes. The files on the punch cards, magnetic tape, printers and typewriters can have only series organization, for the operating principle of such devices is purely series. For example, it is possible to read the fifth recording from the magnetic tape after the preceding four entries are read. Let us note the basic features of the series organized files on the various units.

The input punch card file can be made up only of unblocked fixed-length entries, and this corresponds to the operating principle of the punch card reader. The files which are output to the punch cards or to the printers

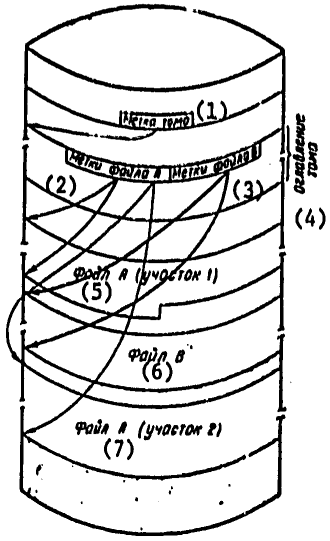

Figure 14. Placement of files on discs

Key:
1. Volume mark
2. File marks of file A
3. File marks of file B
4. Volume index
5. File A (section 1)
6. File B
7. File A (section 2)

must be made up of unblocked entries of any format (fixed length, variable length or undefined length), but no more than 80 bytes for the punch cards and no more than the length of a row for the printers. The files on the typewriter can be made up of unblocked fixed-length or undefined-length entries.

The files on the magnetic tapes are input and output files, and they permit a great variety of formats of the entries: the entries can be fixed length or variable length and undefined length. The fixed and variable-length entries can be both blocked and unblocked. In addition to the input and output files there can be working files on the magnetic tape. They are used as intermediate memory and serve to expand the basic memory. The working files exist only temporarily, and on completion of the program they are discarded as unnecessary. The operating principle of working with the working files is distinguished from the operating principle with the input and output files. For example, if the file is described as an input file, the logical system forbids its use as an output file, that is, it is impossible to write anything in this file; analogously, if it is an output file, it cannot be used as an input file, that is, it cannot be read. The working

files operate on input and output and in contrast to the input and output files on the magnetic tape can be made up only of unblocked fixed length or undefined length entries.

The files on discs can also have series organization. The operating principle of the series organized file or disc does not differ from the processing of the file on magnetic tape; in this case the discs are used as fast magnetic tape. The files on discs are also input and output, and they can be made up of blocked or unblocked entries of any admissible format (fixed length, variable length and undefined length). The working files for the discs have the same purpose as the working files for the magnetic tape.

In contrast to the magnetic tape which is a purely series access device, the physical organization of the disc is such that they permit a great variety of data access methods.

The package of discs is a set of numbered cylinders, each of which can be found by indicating its address (number). To do this no series sorting of the remaining cylinders is necessary. In other words, the search for the cylinder is realized directly by the address. Each cylinder is a set of tracks addressed with respect to the cylinder (Fig 15). On the tracks there are entries, and each entry has its own address which consists of the cylinder address, the track address on the cylinder and the number of this entry on the track. Any entry can be found by its address without sorting all the remaining entries. The disc unit is considered to be a direct-access unit primarily because access to the information on the disc is realized directly by the entry address. The address of each entry is always physically located together with the data following the entry. In addition to the address, the entry can have a key which is in the form of the data perceived by the device as an attribute by which it is possible to recognize the entry. The disc unit also provides for search for the increase by the key on one track or on several tracks.

Fig 16 shows the arrangement of the entries on the track; the structure of the entry containing the key and not having a key is indicated here.

The entries of a series-organized file have no keys, and they need not, for the entries are processed in series one after the other. If the key is a mandatory element of the entry on the disc, the entry address is present always even in the case where the entry has a key. With series access the programmer in practice shows no concern for the address of the entry on the disc -- the logical system itself forms the disc addresses for any reference to the device. For the series file on the discs the programmer determines the section in the packet indicating the tracks and cylinders on which the file is stored. The entries of the series file are located on the tracks of the assigned section one after the other. When all of the entries of one track are processed, the logical system itself proceeds with the processing of the entries on the next track, and so on until the end of file is detected.

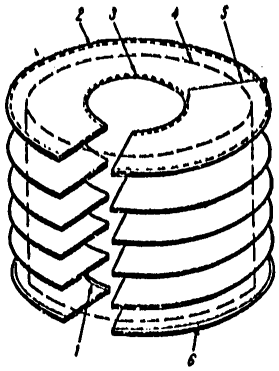

Figure 15. Package of discs
1 -- track 9 of cylinder 202; 2 -- track 0 of cylinder 0;
3 -- track 0 of cylinder 202; 4 -- one cylinder (contains
10 tracks); 5 -- 203 cylinders; 6 -- track 9 cylinder 0

The series access method does not make use of the basic advantage of disc units -- the possibility of access to the entry directly by address. However, it is impossible to state that it is unreasonable to use discs for series files. In many cases this is entirely justifiable and gives good results, especially when all or the majority of the file entries must be processed. By comparison with the magnetic tape the disc insures significant acceleration of the processing and used to process the frequently used series files. For example, the DOS YeS system files are series files, and as a rule, they are located on the disc.

The peculiarities of the storage units on discs, and above all, the possibility of access to the entry directly by its address, are taken into account by the direct and the index-series access methods.

A file with direct access is made up of unblocked entries of any admissible format. The blocking of the entries in practice is meaningless, for the purpose of the method is to insure direct access to each entry. The application of the switches is not mandatory, but they are frequently used. If the keys are not used, then during formation and processing of the file, the actual entry address is indicated, and the logical input-output control system finds the entry by its address. In the case of using keys the programmer will obtain great freedom of action: he can indicate the absolute address of the track and then indicate the key by which the logical system finds the required entry on the track.

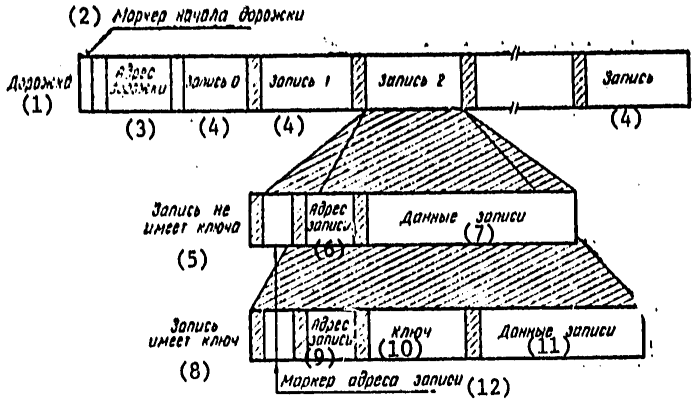

Figure 16. Placement of the entries on the track (cross-hatched sections -- interval between information)

Key:

1. Track
2. Beginning of track marker
3. Track address
4. Entry ...
5. Entry does not have a key
6. Address of entry
7. Entry data
8. Entry has a key
9. Entry address
10. Key
11. Entry data
12. Entry address marker

The file organized by the direct access method has the peculiarity that its entries can be processed in arbitrary order. For example, the eighth entry with respect to order on the disc can be read and processed after processing the 15th or any other entry of the file. Here the time for obtaining the eighth and the 15th entries is approximately identical, and it depends only on the physical arrangement of the read-write heads of the disc at the time of referencing the specific address.

The basic advantage of the direct method can be considered to be the fact that the entries can be processed selectively, and this is especially convenient when performing operations of the type of obtaining information from the general reference file. At the same time no one interferes with the processing of the file entries in series one after the other when large-scale organization of the file is to be done which affects almost all of the entries in the file.

The system does not establish any rules for determining the file structure or arrangement on the disc. The programmer himself determines these characteristics for his own file and must himself be concerned for how the true address of the entry in its file will be determine. If the file is organized by the direct access method, then in order to plan the entry in the file the problem program must indicate the address of this entry on the disc. In this case the address can be absolute, and this means that the

problem program must indicate the cylinder number for the entry, the track number and the number of this entry on the track. In addition to finding the entry by the absolute address the direct method provides for finding the entry on the track by its key. This method of finding the entry presupposes that all of the entries in the file have a key and it is known which entries are on which track.

The direct access method insures high data processing efficiency and is a universal means for working with files on discs. However, this method charges the programmer with all of the responsibility for determining the address of the entry on the disc, and the system takes in itself only the execution of the functions with respect to reading and writing the data on the disc, relieving the programmer of the necessity for writing drivers for the discs on a physical level.

The determination of the address of an entry on discs which is the basic concern of the programmer working with a randomly organized file is in some cases quite difficult. Actually, if it is a large file, it is necessary to apply definite effort to know the absolute address of each entry or know at least which entries are on which track (in the case where the entry is found by a key). The problem of determining the address of an entry is taken off the programmer by the index-series method of access. This method presupposes index-series organization of the file on the disc in which the placement of the entries in the file is dependent on the entry key. All of the entries of the index-series file must have a key; they can only be of fixed length, blocked or unblocked.

In order to create a file with index-series organization, it must first be sorted on the entry key. When a file sorted on the entry key is written on the disc, the logical system constructs an index for it which contains information about the arrangement of the entries on the disc. There is a track index and a cylinder index. The track index is made up of entries -- track indicators. For each track of the file there is an indicator which contains the track address and the highest key of the entries located on the track. Analogously, the cylinder index is made up of entries -- cylinder indicators. For each cylinder of the file there is one indicator which contains the cylinder address and the highest key of the entries located on the cylinder. If the user wishes, a third, principal index can be constructed. It is made up of entries -- the track indicators of the cylinder index. For each track of the cylinder index there is an indicator which contains the track address and the highest key contained in the track indicators of the cylinder index (Fig 17). It is recommended to have a main index for the large file, the cylinder index of which occupies many tracks. The entire operation with respect to the formation of the indexes is performed by the system itself when creating the file.

The index-series method of access permits processing of the file entries both selectively, similarly to the direct access method, and in series. Both methods are organically combined. If the entries are processed selectively, the key of the required entry is communicated to the system.

The system itself finds the entries in the file, using the indexes. It examines the cylinder index to determine the cylinder on the tracks of which the entry is located. When the cylinder is determined, the track index is examined in order to find the cylinder track containing the entry and only then is the entry found on the specific track of the cylinder. This method of finding the entries resembles a direct search although the technique for executing it differs from the technique used in the direct method.

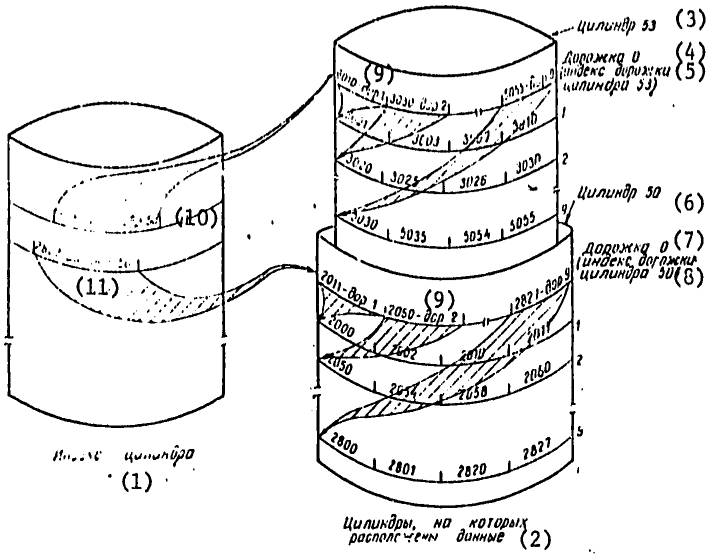

Figure 17. Example of an index-series file

Key:

1. Cylinder index
2. Cylinders on which data are located
3. Cylinder 53
4. Track 0
5. (Track index of cylinder 53)
6. Cylinder 50
7. Track 0
8. (Track index of cylinder 50)
9. Track ...
10. Cylinder 53
11. Cylinder 50

The entries of the index-series file can be processed in series. This does not mean that it is necessary to process the entire file in series beginning with the first entry to its end. The index-series method is convenient in that it can give a section of the file all of the entries of which will be processed in series. Here the beginning of the section is determined directly, using the indexes, and then all of the entries of this section are sorted in series. The indication of the beginning and end of the section processed

in series is given to the logical system by the problem program. This processing procedure where the speed of random search and the ease of indicating the required entry are combined is especially conveniently used when working with large files which are used for more than one application. For example, it is possible to construct an index-series file containing information about the library of an institution and use it both to obtain information about individual books in the library and to obtain information about a defined topic. The reference regarding a specific book obviously will be obtained by using direct access to the required entry, and in order to obtain a list of books on a defined topic, both selective and series processing are required.

A distinguished feature of the index-series organization is also the fact that when creating the file on the disc, a preference zone is isolated which is designed for the addition of new entries to the file. This permits us to avoid copying the entire file which usually is required when adding entries to the file having series organization. Although the added entries physically will not be arranged in a series by the key, they will be processed in the required series.

The index-series method is very convenient for operation: it combines the possibilities of both the direct and series methods. This method presupposes the greatest conveniences for the programmer, for only the entry key is required of it, and no information is needed about the address of the entry on the disc. The system itself quickly finds the entry, knowing only its key. However, it must be noted that the index-series organization of the data somewhat decreases the useful value of the disc memory as a result of the overflow indexes and regions. In addition, for the selection of each entry it is necessary to make at least two references to the disc, and this somewhat increases the access time, for example, by comparison with the direct method.

Functions of the Logical Input-Output Control System. The basic goal of the logical system, independently of the access method, is making the logical entries from the input files available to the problem program and placement of the logical entries in the output files. This goal is not so simple as it can appear at first glance. It consists not only of compiling and starting the driver for the corresponding device. In addition to the fact that the logical system actually provides the drivers for all of the input-output devices of the YeS EVM system, freeing the programmer of this work, it takes into account a variety of versions which occur from the combination of the parameters of the device and the file which is on the device and the requirements of the program. The functions, the execution of which it takes on itself, are realized by the system in practice most effectively. Thus, the system takes into account the possibility of the simultaneous execution of the input-output operations with the processing and makes available various methods insuring this combination to the problem program.

In performing the basic goal with respect to the delivery of the logical entries, the logical input-output control system realizes many functions adjacent to it, relieving the programmer of the majority of concerns for input-output. The logical control system takes on itself also the execution of such significant functions as the opening and closing of files; blocking of the entries, isolation of an entry from a block (deblocking); the processing of the "end of file" and "end of volume" conditions; servicing of the operations of controlling the input-output devices (for example, controlling the format on a printer).

Opening and Closing Files. In the general case the processing of the file must begin with the opening procedure. The basic purpose of this procedure is to establish what problem program the file belongs to and to permit this program to work with the file. The opening procedure is especially important when processing files on magnetic tapes and on discs, for there are large files on these devices which cannot be recognized visually by the operator. During opening of the files on magnetic tapes and discs, the most significant part of the work is done with respect to organization of the protection of the file from inadvertent attempts to destroy it.

The opening procedure establishes to what problem program the file belongs by using the information such as the file identifier and the registration number of the volume which, on the one hand, is written together with the file and is contained in the mark directly on the carrier, and on the other, the information must be placed by the programmer in the assignment. The opening procedure requires that the identifier of the opening file and the registration number of the volume on which it is contained completely coincide with the analogous characteristics indicated in the assignment. The comparison indicates that the required file has been established, and the problem program is permitted to work with it; when there is no comparison, the system informs the operator and waits for additional instructions from him. Until the instructions are received from the operator no operation with this file is permitted.

The described method controls the ownership by the problem program of primarily the input files. For the output files it is primarily important to establish whether the carrier is free on which the file is to be written. Any file can be located on an established carrier which possibly is not needed and then it can be destroyed, permitting a new file to be written in its place or, in the opposite case, it is impossible to give this permission. The answer to the question of whether the carrier can be used is primarily obtained by checking the storage time of the file written on the carrier. In order to make this check the opening programs read the mark of the old file in which it is indicated to what time the file cannot be destroyed. If the file storage time has not expired, the system informs the operator of an attempt to destroy the existing file, and it does not permit the new file to be created on the indicated carrier without his explicit instructions.

In the case where the carrier contains no file or the storage time of the old file has expired, the opening programs permit the creation of a new file on this carrier and begin this creation by writing the standard mark for a file.

The logical system can execute the described checks only in the case where the files have marks. If the files do not have marks, no checks can be made at opening time.

The input-output control system takes on itself the basic work with respect to the creation and checking of the system file marks. The programmer must only deliver information to the system on the marks of its file such as the name of the file, the data of expiration of the file storage time; the series number of the volume if the file occupies several volumes; the series number of the file on the indicated tape if it contains several files; the organization attribute (series, direct, index-series) of the file on the disc; the sections occupied by the file on the disc. The information about the file marks is delivered by the programmer with the help of the controlling operators in the assignment for execution of the problem program. This information is received by the assignment control program, it is edited, and the mark is placed on the cylinder from which in the future on execution of the file opening procedure, the input-output control system extracts and reduces it.

After the file is opened, the system permits it to be processed until the problem program communicates to the system that the file must be closed. The request to close the file indicates that the work with the file is ended, and consequently, the logical systems must forbid work with it, executing the procedures with respect to closing the file. The basic purpose of the closing procedure is the following: for a file on magnetic tape -- processing and creation of the terminal marks and forbidding the use of this file; for the file discs, this, in turn, is forbidding the use of the file and in some cases, freeing of the position on the disc which this file has occupied. The position on the disc is freed if the file has been defined as a working file.

The procedures for opening and closing the files are applicable to any file independently of the organization and the type of device; therefore the programs of the logical input-output control system realizing these procedures are transits and are executed in the transit region of the controlling program.

The user can expand the functions performed by the logical input-output control system during process of opening and closing the file as a result of connecting to the system procedures for opening and closing its own subroutine. By using this subroutine the user can create his own marks for the file (user marks) to supplement the system marks and then use this information at his discretion, for example, for organization of additional checks.

If the file has user marks, and there is a particular subroutine for processing them in the problem program, the programmer must communicate their presence to the logical system. The programmer must make this indication during a description of the file. The system programs for opening and closing the file organize output to the subroutines for processing the user marks which is contained in the problem program. Here the system opening programs for the input files read the user marks from the magnetic tape or disc and transmit them to the problem program for processing, and for the output files the user marks are obtained from the problem program, and they are written on the magnetic tape or disc. The system closing programs proceed analogously, reading the terminal marks of the user for the input files and writing the terminal marks of the user for the output files formed in the problem program.

There is a difference in how the programmer feeds the information to the system for the creation and checking of system marks and user marks. The information for the system marks is fed during the assignment and is stored by the system on the mark cylinder. The information for the user marks is not supplied during the assignment; it is not on the mark cylinder. This information usually is contained only in the problem program itself.

The logical input-output control system also allows for the case where the file on the magnetic tape does not contain any marks, and this means that the system opening and closing procedures for this file in practice are not operating, that is, they do not execute any checks. This is justified, for example, for working files.

Another admissible case is the case where the file does not contain standard marks, but has nonstandard user marks. In this case the system opening and closing procedures do not in practice always operate. They only organize the output to the problem program which itself is obligated to perform the operations with respect to processing the nonstandard marks, including their reading and writing on magnetic tape.

In all cases where the file on magnetic tape has marks, a region must be isolated in the problem program region which is used as the working region during the formation of the reading of the mark from the carrier during execution of the opening and closing procedures. The indication of isolation of a region for the mark is realized by the programmer by using a special controlling operator in the work assignment.

The input-output region for the file marks is allocated in the problem program in the case where the file has direct or index-series organization. The series organized file on the disc does not require that the problem program allot an input-output region for the marks, for there is a region in the opening and closing system programs themselves for marking this file.

Blocking and Deblocking the Entries. The logical input-output control system operates with the data entries. For such devices as punch card units or printers, the size of the logical entry, as a rule, coincides with the size of the physical entry (with the size of the punch card or the row on the printer).

The magnetic tapes and discs do not fix the size of the physical entry in advance; these devices permit a sufficiently broad range of sizes of physical entries (from 18 symbols to the maximum size of the basic memory). The size of the physical entry, that is, the entries which are sent by one input-output operation influences the operating efficiency and the useful volume of the magnetic tape or disc. For example, if all of the physical entries on the magnetic tape contain 80 symbols each, then as a result of the spaces between entries the usable volume of the magnetic tape for the YeS-5010 unit is approximately only 10% of the total length; if the size of the physical entries is increased to 800 symbols, then the usable volume of the magnetic tape increases by 4 to 5 times. In addition, if the size of the physical entries is small, then when processing the file a larger number of references is required to execute the input-output operations.

Taking these factors into account, the logical input-output control system presupposes the possibility of working with the units which includes several logical entries. The unit is a physical entry. The logical system takes in itself the realization of the blocking of the entries, that is, the combination of several logical entries and one block and the deblocking of the entries, that is, the separation of logical entries from the block. For the problem program the nature of processing the block entries varies insignificantly by comparison with the processing of the unblocked entries.

The problem program must take into account several factors pertaining to the entry blocking function. Primarily the problem program must isolate the input-output region in the basis memory sufficient to place not one entry in it, but all of the entries entering into the block. The programmer must decide what procedure will be used to reference each individual logical entry of the block. When several logical entries are combined into one block, the logical system must see that each of the logical entries is now in processing. The logical system presents four methods of solving this problem.

The first method consists in the fact that the logical system places the address of the location of the entry in the input-output region in the register indicated by the problem program. When using this address, the problem program can process the entry in the input-output region itself.

With the second method the problem program, in addition to the input-output region, can isolate the working region which has the size of one logical entry and communicates these to the logical system. The logical system will send each logical entry of the input file into this working zone in series. After the last entry from the input region is sent to the working region, the system reads the next block into the input region from the

external carrier, for the input region is already free. Thus, the matching of the processing of the last entry of each block with the execution of the read operation of the next block from the external carrier will be achieved, and this will permit reduction of the total execution time of the entire problem program. The analogous situation is also possible on output. In the case where the entries are processed directly in the input-output region there will be no matching of the execution of the process and the input-output operation for the problem program.

In the third method the entries are processed in the input-output region, but the problem program has two input-output regions. The system organizes the switching from one region to the other any time that all of the entries in one input-output region have been processed. When one of the regions is free, the inpout-output region is started by using the freed region. In this procedure, greater matching of the processing and the input-output operations is achieved than in the case of using one input-output region and the working region, but then the processing procedure using the two input-output regions requires greater expenditures of the basic memory.

The fourth procedure permits the use of two input-output regions and a working region. The system sends each logical entry in series to the operating region first from one region and then from the other, switching from one region to the other at any time that all of the entries from one input-output region have been processed. Usually this procedure is rarely used inasmuch as it permits time to be saved only when the majority of entries in general do not participate in the processing or with processing is too insignificant.

Use of the Logical Input-Output Control System in the Problem Program. The logical input-output control system is the next link between the problem program and the physical input-output control system in the part which belongs to the input-output programming. The programs with the logical system in the greater part (only the opening and closing transits are an exception) are included in the problem program as finished, previously programmed pieces realizing the fully defined input-output procedures. Being included in the problem program, they become an inseparable part of it and are present in it during the time of existence of this program.

The logical input-output control system is the apparatus for the programmer writing his programs in the Assembler language. During programming on the Assembler, the necssity arises for planning the input-output procedures; determination of the organization of the file, compiling the drivers for realization of the input-output operations, processing unusual situations, realization of the transition from volume to volume in the case of multi-volume files, and so on, that is, the functions which the logical system performs.

During programming in higher level languages than Assembler, these problems do not arise for the programmer in explicit form. The input-output problems

are solved in each of the programming languages by their own means. The programmer running the program, for example, in Cobol, cannot know that the translator from Cobol itself and also the working program constructed by Cobol use the logical input-output control system. The information about this is needed by him, for this is a problem of constructing a specific translator, and there are no possibilities for the programming language.

In the DOS YeS system the logical input-output control system has a universal nature and is widely used in the majority of system programs such as translators, sorting, copy programs and also the working programs created by the translators from the PL/1, RPG and Cobol languages.

In order to use the logical input-output control system, the following conditions must be satisfied:

1. The standard subroutines of the logical system must be located in the program library.

2. The programmer must assign the characteristic of his file which will be perceived by the standard programs as their parameter.

3. Any time it is necessary to request the required functions with respect to input-output, the programmer must reference the standard programs of the logical system.

The logical input-output control system has a tabular-modular structure: the file characteristics are stored by the system in the form of file description tables, and the standard programs realizing the input-output functions are the program modules. Between the program module and the table of description of the corresponding file there is a close relation: the data of the file description table are parameters for the program module. In the system there are independent modules and description tables for each type of device and each method of accessing the data. Thus, for series access the logical system provides for independent modules and rules for constructing the file description tables on punch cards, a printer, magnetic tapes and discs. For the direct-access method in the system there is its own module and the file description table corresponding to it on the disc constructed by the direct access rules. Analogously, for the index-series access there is an original description table and modules oriented toward processing of the file with index-series organization.

The logical system module is oriented to the type of device and the method of access to the file on this device. Here the module insures execution of all the functions which can be needed for processing the organized file on a device of this specific type in the corresponding way. In order to emphasize the universalness and completeness of encompassing all possible situations, the logical system module will be called a generalized module. The generalized module, for example, for a file on magnetic tape (the

series organized file) insures execution of such functions as reading and writing the blocks, skipping zones and rewinding the magnetic tape, blocking and deblocking the logical entries, the processing of erroneous situations and certain other functions. For files on discs there are generalized modules which realize the functions of series, direct and index-series access. For convenience of working with the system files which can be formed on the devices of several types (for example, on magnetic tapes or on discs or on punch cards) the logical system presents a special generalized module. It takes into account the specific nature of organization of the files on the SYSRDR, the SYSIPT, SYSPCH, SYSLST logical units, and it permits the use of any of the physical units which can be assigned to these system logical units for the files connected with these units without many changes.

The generalized modules of the logical input-output control system are permanently stored in the library of initial modules in the form of macro-definitions written in the Assembler macrolanguage. From the generalized module the system can obtain a specific module taking into account the requirements of processing the files for the specific problem program.

The programmer describes the module which is needed for processing its files, indicating the parameters in the corresponding macroinstructions available in the system, describing the modules. For each generalized module there is its own macroinstruction describing this module. The set of macroinstructions for describing the modules is presented in Table 5.

Table 5

List of Macroinstructions for Describing the Modules

Macroinstructions	Purpose
CDMOD	Module for files on punch cards
MTMOD	Module for files on magnetic tape
PRMOD	Module for files on printers
PTMOD	Module for files on punch tapes
DIMOD	Module for files which do not depend on the type of device
SDMOD	Module for series organized files on discs
DAMOD	Module for direct access method for files on discs
ISMOD	Module for index-series files on discs

The input-output modules can be included in the problem program in two ways: in the translation step and in the editing step. When writing the program in the Assembler, the macroinstructions for description of the modules can be included directly in the problem program. In the process of translating the problem program, instead of the macroinstructions for description of the modules it can include the specific modules adjusted in accordance with the requested functions.

Another method of including the input-output modules presupposes that the specific module is prepared (translated with specific parameters) independently of the problem program, and it is placed in the library of objective modules. From it the module can be included in any problem program in the editing step. The procedure for including the modules of the logical system in the editing step is preferable to the first procedure (inclusion of the modules and the translation step), it is more convenient and saves the translation time of the problem program. Actually, the separately translated module can be used in any problem program where it is applicable. The specific module is quite universal; it can process the files having different length of the logical entries, different input-output regions, a different entry blocking coefficient, and so on. Therefore the specific module can be successfully used both in one program for the processing of several files of the same type and in different programs.

During generation of the system translated modules of the logical system are placed in the library of target modules for each device and all access methods, and these modules provide for processing of the files having different characteristics. For example, in the library of target modules for magnetic tape there are 14 modules, among which is the module which processes only files with blocked or unblocked fixed-length entries; the module which processes files only with undefined-length entries; the module which processes files only with blocked or unblocked variable-length entries; the module which processes only the working files, and so on. In practice cases rarely appear which require new input-output modules although there is always the possibility of obtaining a specific input-output module in accordance with the required parameters.

Each module of the logical input-output control system placed in the library of target modules is assigned a standard name. It is used in cases where it is necessary that corresponding standard module available in the library be included in the program at the editing time. The standard input-output modules are used, for example, in the working programs constructed by the DOS YeS translators.

In the problem program for each file, in addition to the input-output module, there must be a file description table matched with the used module. In the file description table the parameters must be indicated which the module correctly perceives and processes. For each generalized module there is an independent file description table which permits use of the standard method to indicate the characteristics of any file of a specific type. For example, for a file on magnetic tape using the file description table it is possible to indicate the file characteristics such as the name of the file, the type of file (input, output, working), the presence of marks, the entry format (fixed-length, variable-length or undefined length), the length of entry, the input-output region and size, the error processing regime, the magnetic tape rewind regime, and so on.

The file characteristics are indicated by the programmer by using the macroinstructions for describing the file. The logical input-output control system constructs the description table based on the macroinstruction parameters in the required format. The set of macroinstructions for description of the file is presented in Table 6.

Similarly to how the input-output modules are included, the file description tables can be included in the problem program by two methods: in the translation step and in the editing step. The programmer can use the macroinstructions for description of the file in his program, and then the corresponding tables will be included in his program during translation by the Assembler.

Table 6

List of File Description Macroinstructions

Macroinstruction	Purpose
DTFCD	Description of file on punch cards
DTFMT	Description of file on magnetic tape
DTFCN	Description of file on typewriter
DTFPR	Description of file on printer
DTFPT	Description of file on punch tape
DTFDI	Description of file independently of type of device
DTFSD	Description of series-organized file on discs
DTFDA	Description of file on disc having direct organization
DTFIS	Description of index-series file on discs
DTFPH	Description of file processed on the physical level using the opening-closing procedures

With the other procedures the file description macroinstructions are formed as individual modules, they are translated independently and are placed in the library of target modules, and in the editing step they are included in the problem program. As a rule, the file description tables are included in the problem program in the translation step, for this is more natural for the programmer. Actually, the file description macroinstructions are convenient to have in the text of its own program: in the macroinstructions the characteristics are indicated such as the symbolic addresses of the input-output regions, the subroutines for processing the user marks, the subroutines for nonstandard reaction to erroneous directions, and so on, that is, the information which is specific to the specific program and the specific file.

In contrast to the input-output modules in the library of target modules there are no standard file description tables.

For each file processed by the program its own file description table must be constructed independently of whether the created table is placed in the library or not. For example, the DOS YeS translators include the file description tables in the target program when constructing the working programs in the format required by the logical system.

Thus, in the problem program the file is described, and there is a logical system module designed to process this file. Each time when the problem program requires the execution of defined operations on the file, it is necessary to reference the module with the corresponding request. In order to send the request to the module the programmer writing his program in Assembler, has a set of macroinstructions. In contrast to the declarative macroinstructions describing the files and modules, the macroinstructions requesting operation of the file are called imperative.

The imperative macroinstructions, just as the declarative ones, are separated with respect to methods of access and types of devices. Some of the imperative macroinstructions are applicable to several of the access methods and several types of devices; others are specific to the specific types of devices and methods of access. For example, the OPEN and CLOSE macroinstructions requesting opening and closing of the files can be used for all of the access methods and for all types of devices except the typewriter. The GET and PUT macroinstructions used to obtain the logical recording from the input file (GET) and put the entry in the output file (PUT) are designed for processing of the file having series or index-series organization. There is a group of instructions which are applicable only for processing the index-series file.

Each access method corresponds to the group of imperative macroinstructions. Although the means of the macroinstructions performing such functions for different methods coincide, the macroinstructions strictly oriented to the method are distinguished by the parameters and functioning. For example, the READ macroinstruction has different format and executes the reading of the entries differently when processing the series, direct and index-series file. The list of imperative macroinstructions as applied to the access methods and types of devices is presented in Table 7. The symbol "X" is used to denote the macroinstructions which can be applied for processing the file on a specific device with corresponding organization.

Table 7

List of Imperative Macroinstructions

(1) Макрокоманда	(2) Последовательный метод доступа: 3 Пишущая машинка	4 Печат.	5 Ввод перфокарт	6 Вывод перфокарт	7 Магнитная лента	8 Диск	9 Ввод перфоленты	10 Вывод перфоленты	11 Прямой метод доступа	12 Индексно-последовательный метод доступа	Назначение (13)
OPEN OPENR		×	×	×	×	×	×	×	×	×	Opens the file
CLOSE CLOSER		×	×	×	×	×	×	×	×	×	Closes the file
FEOV					×						Establishes the "end of volume" condition for the magnetic tape
LBRET						×	×			×		Returns control to the logical system from the subroutine for processing the user marks or nonstandard marks
GET	×		×			×	×	×			×	Generates the next logical entry of the file for processing
PUT	×	×		×		×	×		×		×	Outputs the indicated logical entry
RELSE						×	×					Indicates the necessity for transmitting entries available in the input region and proceeding to the processing of the entry of the next block
TRUNC						×	×					Permits writing of the partially filled block
CNTRL		×		×		×	×			×		Executes the control operations: rewind, skipping of sums, skipping of rows on the printer, and so on
PRTOV		×										Controls the movement of the paper on the printer
READ						×	×			×	×	Starts the operating of reading data into the input region
WRITE						×	×			×	×	Starts the operation of writing the data from the output region
CHECK						×	×					Causes waiting for completion of the started input-output operation on the READ, WRITE macroinstructions

Key: 1. macroinstruction; 2. series method of access; 3. typewriter; 4. printer; 5. punch card input; 6. punch card output; 7. magnetic tape; 8. disc; 9. punch tape input; 10. punch tape output; 11. direct access method; 12. index-series method of access; 13. purpose

[Table 7, continued]

(1) Макрокоманда	(2) Последовательный метод доступа: 3 Печатная машинка	4 Печать	5 Ввод перфокарт	6 Вывод перфокарт	7 Магнитная лента	8 Диск	9 Ввод перфоленты	10 Вывод перфоленты	11 Прямой метод доступа	12 Индексно-последовательный метод доступа	(13) Назначение
POINTS					×	×					Establishes the working file at its beginning
POINTR					×	×					Establishes the working file in the position for subsequent reading of the indicated entry
POINTW					×	×					Establishes the working file in the position for subsequent writing of the indicated entry
NOTE					×	×					Indicates the necessity for placing the entry identifier in register 1; for tape this is the number of written or read blocks; for a disc, this is the disc address of the written or read entry
WAITF									×	×	Causes waiting until the started input-output operation is completed with respect to the READ, WRITE macro-instructions
SETFL										×	Used for the creation or expansion of the index-series file and performs ready operations
ENDFL										×	Completes index-series file loading regime established by the SETFL macro-instruction
SETL										×	Establishes the write address, beginning with which the index-series file will be processed in series
ESETL										×	Causes completion of the series processing started by the SETL macroinstruction

[See key on p 91]

CHAPTER 4. MEANS OF DEVELOPING PROGRAMS

The means of developing programs play an important role in the operating system: the output capacity of the computer depends on how efficiently the programs can be developed and how completely they make use of the capabilities of the operating system.

The DOS YeS operating system makes a large set of means available to the programmer, which makes it possible to automate the entire process of setting up the problem on the computer.

The process of setting up a problem on the computer can be represented as series execution of the following operations: planning of the program structure; preparation of the program for execution; execution of the program.

Planning the Program Structure

The planning of the program structure and encoding it in the initial programming languages are a step in which the entire meaningful load is borne by the programmer. In this step the programmer selects language means for his problem, and this choice determines what system capabilities he will use.

The programmer must plan the organization of the program before the program is written. Many factors can influence the choice of program structure: the habits of the programmer based on previous experiences, the capabilities of the operating system, the proposed basic memory size, the programmer liability problems. In addition, the structure of a large program and its preparation are significantly influenced by such factors as the necessity for distributing the work among the various programmers. In turn, the distribution of the operations among the participants of the collective or what is equivalent, the separation of the large problem into smaller meaningful parts are influenced by the capabilities of the operating system.

In the DOS YeS system there are general rules, the use of which in practice does not depend on the choice of language for encoding the particular program. These system means can include the means for combining the parts

of the program, use of the finished programs from the libraries, the rules for formating the assignments for execution.

In the DOS YeS system the general system for preparing the program includes two basic steps: translation of the initial program written in any of the available programming languages and editing of the target modules obtained as a result of translation. This system is valid for any DOS YeS translator; the structure of the target module is always identical and does not depend on which translator creates the module. This standardization of the translation results permits combination of the target modules obtained by various translators into one program in the editing phase.

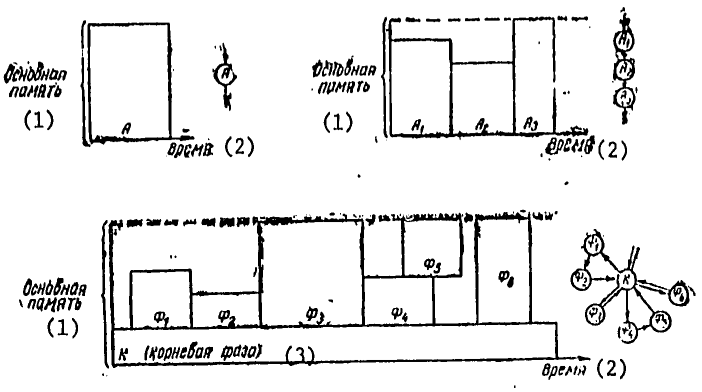

Figure 18. Occupancy of the basic memory and flow charts for the programs of different structure: upper left -- simple structure; upper right -- structure with overlay without the base phase; bottom -- structure with overlay with base phase

Key:
1. Basic memory
2. Time
3. Base phase

Thus, the adopted stepped preparation of the programs makes it possible for the programmer to separate his problem into parts and select the most appropriate programming language for each part. The system provides for independent translation of each autonomously prepared part and combination of these parts into one program of the required structure.

For his problem the programmer can plan a program of one of three structures (Fig 18):

Simple structure where the entire program is one phase and is called into the basic memory as a whole for execution;

APPROVED FOR RELEASE: 2007/02/09: CIA-RDP82-00850R000100060066-2

The structure with overlay with basic phase where the program is made up of several phases; here the base phase is present in the basic memory during the entire program execution time, and the remaining phases are called into the basic memory by the base phase alternately to one and the same place;

The structure with overlay without base phase where the program is made up of several phases in the basic memory occupied a section equal to the length of the largest of the phases; in this case in the memory only one of the phases is permanently present, and each executed phase calls the next one into its place, destroying itself.

In planning the program structure, the programmer always has in mind the fact that each part of his problem is encoded in the form of one or several initial modules which can be autonomously translated, stored in the system in the form of target modules and after editing one or several phases obtained which represent one program. Inside the phase subroutines can be planned which also can be independently included, they can be obtained in the form of target modules and then included in the phase in the editing step. In addition, the programmer can include finish subroutines in his program which are contained in the form of target modules in the system or other libraries of target modules.

Thus, when writing the program the programmer can proceed as follows:

Divide the program into segments;

Isolate the subroutines inside the segments;

Use the finished subroutines of the system contained in the libraries;

In the initial programming languages determine the relations between the subroutines inside one segment, between the different segments and also the interaction between the subroutines and the segments in the process of execution of the program.

Let us illustrate the enumerated possibilities in the example of writing a program in Assembler. Let us propose that the program has the structure depicted in Fig 19, that is, it is made up of five segments, each of which is ready for execution in the program, and is a phase.

Phase A is the base phase and remains in the basic memory during the entire execution of the program. Phases B and E occupy the same region of the base memory, but during different time periods of execution of the program. Thus, in our example, it must take into account, for example, the relations between phases:

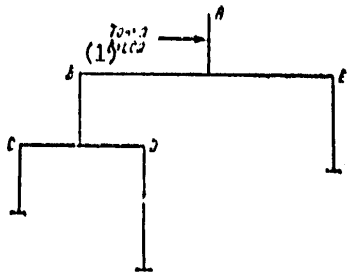

Figure 19. Example of a program with overlay

Key:

1. Input point

The phases C and D are independent of each other, they cannot be present simultaneously in the basic memory, control is obtained under defined conditions from phase B and can have information relations with phases B and A;

Phases B and E are also independent of each other; they receive control from phase A and can only be informationwise dependent on it;

The execution of the program begins with phase A.

In order to determine the relations between the individually translated parts of the program -- the initial modules -- there is an apparatus of external relations in Assembler which permits us to define the external and internal data and the general regions.

The data which are required for execution of the module are external, but they are determined in another initial module. The external data can be, for example, the address of the entry to the module; the address of the variable formed in the other module.

The output data are the data which are defined in the initial modules, but they are also used in other modules. The data defined as input in a specific module are external for another, and, on the contrary, the data which are external in the specific module must be defined as input in another initial module.

Making use of the input and the external data, the programmer defines the overlay references between the individual modules. For this purpose, there are the following operators in Assembler: ENTRY -- for indication of the input data -- and EXTRN -- for external data.

The individual program modules can exchange information also through the common. The common is a section of a memory defined by the problem program itself for use by all phases of the program. For example, data can be placed in the common which are processed by the different phases. In the Assembler language there is an operator COM for defining common. The analogous means for defining the input, external data and common regions also exist in other programming languages.

During the writing of the program with overlay the programmer, in addition to the instructions for the above-described communications, must define the locations in which the calling of the required phase into the basic memory is to be organized. The communications of this type are also described by the means of the initial programming languages. In Assembler, for example, the macroinstructions FETCH and LOAD are used for this purpose. FETCH is used in cases where it is necessary to load a phase into the basic memory and immediately after loading, transfer control to it, that is, executed directly after loading. The macroinstruction LOAD is used when it is necessary to load the phase into the basic memory, but it is not necessary to execute it directly. After the phase is loaded, control is returned to the calling phase.

The division of the program into segments can be considered as the initial step in planning the program structure. The next step is finding the structure of each individual initial module:

Breakdown of the initial module into subroutines which must be created by the programmer himself;

Inclusion of the finished subroutines from the library in the module;

Establishment of relations between the subroutines of the module;

The relations between subroutines entering into one initial module are called direct relations.

In the Assembler language, direct communications are established between the subroutines using the macroinstructions CALL, SAVE and RETURN. Analogous means for establishing the direct communications exist also in other programming languages.

For the characterization of the communications among the subroutines inside the module, the level concept has been introduced. A program from which the execution of a module begins is called the zero level program or the basic program. The subroutines to which the basic program reference is directed are called the first level subroutines. Analogously, the subroutines to which the first level subroutines make direct references are called the second level subroutines, and so on. Inside the initial module programs of any level are permitted. The same subroutine can be a subroutine of different level depending on the level of the subroutine referencing it.

Fig 20 shows an example of direct communications between the subroutines of the initial module which is made up of the basic program A and two subroutines B and C. Here B is a first level subroutine, C is a second level subroutine.

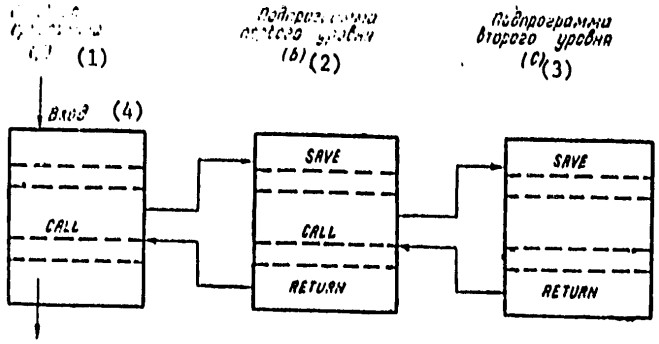

Figure 20. Example of direct communications between subroutines

Key:
1. Basic program (A)
2. First level subroutine (B)
3. Second level subroutine (C)
4. Input

Translation and Editing of the Program

The planning of the program structure and coding it in the initial program languages is the initial step in the process of setting up the problem. In this step the programmer makes use of the system means without referencing the computer. The next steps which are executed on the computer are translation of the initial modules by the corresponding DOS YeS system translators and editing of the target modules obtained during which the program of the required structure is created. The steps in the program preparation are presented in Fig 21: coding, translation and editing.

Let us note the basic capabilities which the system makes available to the programmer in the translation and editing steps.

The initial modules can be stored in the library of initial modules and then used for coding another initial module of the given program and when developing another program.

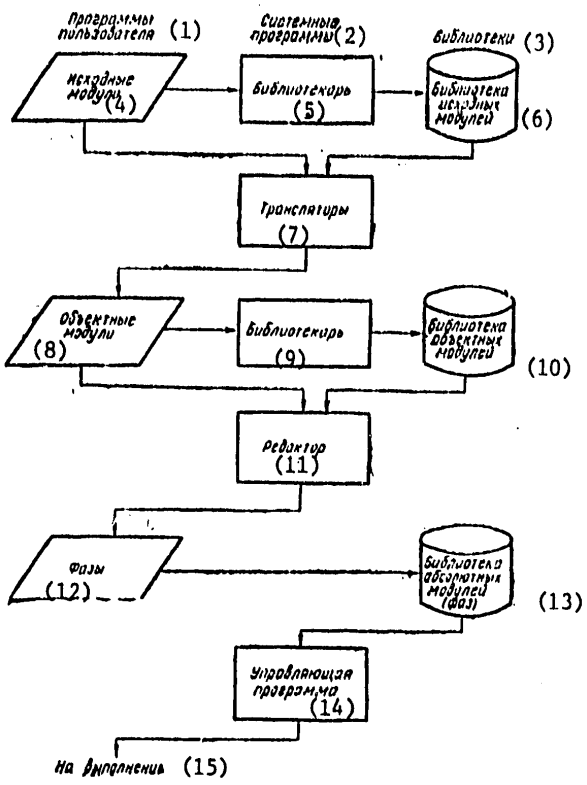

Figure 21. Program preparation

Key:

1. User programs
2. System programs
3. Libraries
4. Initial module
5. Librarian
6. Initial module library
7. Translator
8. Target module
9. Librarian
10. Target module library
11. Editor
12. Phases
13. Library of absolute modules (phases)
14. Control program
15. For execution

There are several methods of working with the target modules. They can be obtained on an external carrier -- punch cards, magnetic tape or discs; it is possible to put the target modules in the library. Directly after translation the target module can go for editing; after editing it leaves no traces in the system. The indicated possibilities of working with target modules create sufficient flexibility when preparing the program. Each part of the program can be prepared autonomously until the

stage of obtaining the target modules, the modules can be obtained independently with respect to time and are used altogether after the last is obtained. In addition, the target modules obtained on the external carrier or fitted in the library can be used when preparing another program.

The target modules must be edited so that a program ready for execution will be obtained from them. In the editing step the programmer describes the structure of this program by means of controlling operator of the Editor program, indicating the division of the program into phases and the composition of each phase, that is, defining the target modules which must enter into the phase; he gives the load address and the point of entry for each of the phases.

After the program is edited, on examination by the programmer it can be executed directly or it can be placed in the library of absolute modules. It is expedient to place the program in the library of absolute modules in the case where execution of it more than once is required in some time period. Each new execution of the program placed in the library of absolute modules does not require preliminary translation and editing, and thus system time is saved. After this program is to be no longer used on the computer, it can be removed from the library of absolute modules.

Program Execution

The editing step completes the process of program preparation, after which it is ready for execution. The next step is the solution of the problem.

In order that the program be able to be executed, an assignment must be prepared for its execution. As is known, each assignment can be made up of one or several assignment steps. It is recommended that certain factors be considered when planning the assignment.

Since the programs are entirely independent of each other, they can be formed as individual assignments or as steps in one assignment. It is expedient to form them as individual assignments, for in the case of abnormal execution of one step of the assignment, the entire assignment is removed.

If two programs are connected only by external files on punch cards, magnetic tape or discs, they usually are formed as steps of a single assignment. For example, the only connecting link between translation and editing is the target module which is always on an external carrier. Therefore, as a rule, translation and editing are steps of one assignment. Another step in this assignment can be the execution of the edited program.

The solution of the problem is realized under the control of the controlling program. By using the information of the controlling operators, the assignment control adjusts the system to the execution of the required program, the Supervisor loads the finish program into the basic memory from the library of absolute modules, puts it into operation in accordance with the priority of the division and then controls its execution to completion.

In the different steps of the problem statement process, various means of the DOS YeS operating system are used. In the program encoding step, programming languages are used; in the translation and editing steps, the translators, the Editor program and the Debugging program. The libraries are constantly used along with the large set of such programs as sorting, the information copy programs, the disc package servicing programs, and the unit checking program.

Programming Languages

The DOS YeS operating system makes available the programming languages oriented to various classes of problems: Assembler -- a machine oriented language; base Fortran and Fortran IV -- for scientific and technical problems; RPG -- for accounting problems; Cobol -- for economic problems; PL/1 -- universal programming language.

Assembler. The Assembler language is a symbolic programming language, and it can be used to solve problems of any type, for it permits direct use of the system of instructions of the YeS EVM. The Assembler language provides the programmer with a convenient method of writing the YeS EVM system instructions. Its basic advantages by comparison with machine languages are the following: the possibility of addressing the program elements by symbolic means, mnemonic indication of instruction codes, description of the constants and the work zones for the data in convenient form. By using Assembler, the programmer has the possibility of isolating the program sections in the initial module, determination of the external relations among modules which are translated individually from each other, and construction of a program of any structure.

The system makes a set of macroinstructions available to the programmer using Assembler, by means of which programming on the physical and logical levels of input-output operations is possible, various functions can be requested from the Supervisor, for example, the time of day, printing out memory, loading phases.

The Assembler language includes the macromeans permitting the application of the macrogeneration method when preparing the program. Macrogeneration is a method of preparing programs based on the previously finished macrodefinitions which are program-like sets of operators. Each macrodefinition is placed in correspondence to a macroinstruction by means of which it is possible to reference the macrodefinition. Actually, the set of macroinstructions is an expansion of the Assembler language.

The system makes a defined set of macroinstructions available to the programmer which realize the system functions and are called system macroinstructions. The set of macroinstructions of Assembler can be expanded by the programmer himself. By using the macromeans of Assembler, the programmer can develop his own macroinstructions and macrodefinitions. The principle of using the macroinstructions developed by the programmer

is analogous to using the system macroinstructions and consists in the following. The programmer writes a macroinstruction in his program using the Assembler language in which the actual parameters are given. The translation process includes the macrogeneration step. In this step Assembler, analyzing the initial program, detects the macroinstruction, finds the macrodefinition corresponding to this macroinstruction in the library of initial modules and adjusts the macrodefinition in accordance with the actual macroinstruction parameters. As a result of this adjustment, the set of operators of the initial language is formed called the macroexpansion which is included in the initial program at the point of calling its macroinstruction.

In the DOS YeS system there are two translators from the Assembler language: Assembler E and Assembler F. Both translators have the same input language; any program in Assembler can be translated by one translator or another. The translators are distinguished by their operating speed and the size of the basic memory required for execution of the translation. Assembler E requires 14K bytes (K=1024) of the basic memory for its operation; Assembler F requires 44K bytes, but the latter is faster than Assembler E.

Both translators are executed only in the background division; as a result of the translation, the target module is created.

RPG. The RPG language is a problem-oriented programming language and is designed for the solution of accounting problems. The content of the accounting problems basically is exhausted by the following processes: the keeping of the files (creation, correction, renewal), the compilation and printing of the documents: tables, lists, summaries, reports. An insignificant proportion of the calculations and a large input-output volume are characteristic of such problems.

The RPG language does not require that the user have special knowledge of the computer. The advantages of this language when solving the data processing problems can also include the fact that the programmer must describe only the problem without describing a detailed algorithm for its solution. For description of the problem in RPG there are several types of forms on which the input and output files and the operations which must be performed on them are described.

The RPG language provides the user with the following basic possibilities:

Introduction of the files having series, arbitrary or index-series organization;

Creation of the various types of printed files and files for other purposes having series, arbitrary and index-series organization;

Performance of the calculations on the data: addition, subtraction, multiplication, division and other processing operations;

Editing and printing each introduced entry and the processing results;

Calculation of the results and printing these results out on satisfaction of the given conditions;

Monitoring the input entries for order of the defined types of entries in the group and for ordering (as the given attributes increase or decrease) of all of the introduced entries of the file;

Selection of the entries of the different files with respect to the defined attributes and execution of compatible processing of the entry data;

Realization of the table search;

Printing out required heading information at the beginning of each new report sheet and some summary information at the end of each sheet;

Making the exit to the subroutines in Assembler.

The RPG translator is executed in the background division and requires 14 bytes of basic memory for its operation. The RPG translator converts the program from the initial language to the target module.

Fortran. Fortran is designed for programming scientific and technical problems. In the DOS YeS system there are translators from basic Fortran and Fortran IV.

The basic Fortran language is distinguished by simplicity, the absence of complex structural elements by comparison with the other algorithmic languages. It is easy to learn and convenient for programming many scientific and technical problems. The language provides the programmer with great possibilities with respect to using the input-output means. The methods of series and direct access to the data are permissible; the data transmission is possible both with and without the format conversion. An important property of the language is the presence of means permitting determination of the relations between the autonomously translatable parts of the program (the subroutines).

The Fortran IV language completely includes the basic Fortran language and makes appreciably more possibilities available to the programmer, for example, such as the possibility of the processing of the logical, complex and sexadecimal data, more flexible means of communication among subroutines, and convenient debugging means.

The translators from the two languages convert the program written in the initial language to the target module and can be executed only in the background division. In the translators provision is also made for the diagnostic means which permit discovery of the syntactic errors in the initial program. The basic Fortran and Fortran IV subroutines which are designed for calculation of the various mathematical functions and

execution of certain service operations are contained in the library of target modules. The reference to the subroutines is contained in the Fortran program, and the subroutines themselves are included in the program in the editing phase of the target module obtained as a result of the translation.

PL/1. The PL/1 language is a universal, highly developed programming language. It can be used to solve both scientific-technical and economic problems.

In the DOS YeS system translator the PL/1 realizes a subset of the PL/1 language in which by comparison with the complete language there is no apparatus for asynchronous execution of the operations of the problem program and processor means.

The PL/1 translator operates in the background division and converts the initial module to the target module. In the library of target modules there are PL/1 subroutines which realize the built-in language functions, and they perform the various service operations. The reference to these subroutines is contained in the initial program in the PL/1 language, and the subroutines themselves are included in the program in the step of editing the target module created by the PL/1 translator.

Cobol. The Cobol language is designed for solving the data processing problems. Cobol is oriented to the processing of large-volume files non-connected with the complex calculation. The characteristic features of Cobol are such properties as clarity of the language, relative machine independence, and self-documentation of the programs.

In the DOS YeS system the Cobol language is based on the American National Standard Cobol, it has a modular structure and is represented in the form of eight functional modules: the language nucleus, the series access, direct access, table processing, sorting, creating reports, segmentation means and a library. The language has means of organizing communications among the individually translatable modules and for debugging on the initial language level.

The Cobol translator translates the program in the initial language to the target module, it requires 54K bytes of basic memory for operation, and it can be executed only in the background division.

The Cobol library is part of the system library of the target modules, and the subroutines from this library are included in the target module obtained as a result of translation, during editing. Subroutines of this library realize individual language operators, the conversion of data from one format to another and certain arithmetic operations and special functions.

System Servicing Programs

The DOS YeS system servicing programs provide for the execution of the functions which are necessary to all or the majority of users, for example, the editing of the programs and placement of them in the library of absolute modules, servicing of the DOS YeS libraries, copying of the information from one carrier to another.

The DOS YeS system includes the following system servicing programs: Editor, Librarian, Debug, copy programs, sorting programs, and the unit checking program.

Editor Program. The basic function of the Editor determing its place in the system is the formation of programs from the target modules created by the DOS YeS translators ready for execution. Editing is the concluding and ever present step in the process of program preparation for execution.

The input information for the Editor is the target modules which are created by the translators from the Assembler, RPG, Fortran, PL/1 and Cobol languages. The Editor processes these target modules and performs program phases from them which are stored on the package of discs in the free part of the region of the absolute module library.

Depending on the given operating conditions of the system, the work of the Editor can be completed or execution of the cataloging procedure for the edited program can be requested. In the former case the edited program is maintained in the library of absolute modules temporarily: either until completion of the execution of the current assignment or until a new execution of the editor is requested in the current assignment. During this time it can be called into the basic memory for execution any time when this is requested by the programmer. When the assignment is over or an instruction to edit the next program is obtained in this assignment, the region in the library of absolute modules which was occupied by the preceding editing program is declared free and the preceding program temporarily placed in the library cannot be read from the library of absolute modules, for it is not available there.

In the case where execution of the cataloging procedure is requested, the edit program is cataloged, that is, it is placed permanently in the library of absolute modules. The cataloging of the program permits it to be loaded into the basic memory at any time, for it is constantly present in the library of absolute modules.

Thus, independently of whether the cataloging regime is given or not, the edited program is placed in the disc package region alloted to the library of absolute modules, and it is called from this region into the basic memory for execution. Therefore it is correct to consider that any program is loaded for execution from the library of absolute modules.

The step nature of preparation of the programs adopted in the DOS YeS system, including the editing process, is a universal and flexible apparatus in the hands of the programmer. The programmer has the possibility of:

Breaking down the program into parts, for each of which the most appropriate of the programming languages available in the system is selected;

Autonomously, independently of each other translating each part;

Editing the target modules of each part obtained and combining them into one program ready for execution.

All of the relations which exist between the various autonomous parts of the program are determined symbolically by the programmer in the initial languages: they are retained as external references in the target modules and they receive their specific values only in the editing step.

From the target modules the Editor can form a program of simple structure or structure with overlay with or without a base phase. The program structure is selected by the programmer and he must plan it in the initial language. The basic memory size is defining when selecting the structure. It can be satisfied for the program. By using the structures with overlay it is possible to decrease the basic memory size required for the program.

The selected program structure must be taken into account by the programmer in the step of writing the program and in the editing step. In the step of writing the programs the programmer himself must be concerned about the connections between the individual parts and the method of transmitting information from one part of the program to another, using the means which are available for these purposes in the initial languages. In the editing step by using the controlling operators the programmer must indicate the information for the editor required to construct the program of the required structure. For example, the following is indicated:

It is necessary to construct the program of simple structure or with overlay;

There is a base phase;

What are the objective modules and in what program phases must they be combined;

The division of basic memory in which the edited program will be executed;

The address of the basic memory for each phase from which it must be loaded for execution.

The target module can be considered by the Editor either as an indivisible processing unit or as a set of smaller parts called program sections. Program sections which enter into a single target module are processed by the editor independently of each other, and they can be placed in the various program phases. The separation of the target module into program sections is realized by the programmer in the initial language, and he does not indicate to the Editor in which program phase one program section or another must be included. For example, when the target module which is made up of three program sections C1, C2, C3, it is possible to construct a program made up of two phases A and B. The sections C1 and C3 can be included in the A phase, and section C2 can process the B phase.

The initial information comes to the Editor from the SYSLNK system logical unit for which the disc is always designated. By the beginning of information of the Editor the target modules subject to editing must be contained in the SYSLNK along with the controlling operators determining the structure of the program.

The target modules can be entered in the SYSLNK by the translators directly after obtaining them as a result of translation or by the Assignment Control program, which copies them in the SYSLNK from the SYSIPT system logical unit. The controlling operators for the Editor are placed in the SYSLNK by the Assignment Control program.

The programmer can request the target modules which are found in the personal or systems library of target modules in the edited program phase.

The Editor has apparatus for automatic inclusion of the target modules from the library (personal or system) in the edited program phase. Automatic inclusion activates in the case where after combination of all the program sections and target modules requested by the programmer in the phase, undefined external references remain. For definition of them the Editor automatically calls the required target modules from the library. The automatic inclusion can be cancelled by request of the programmer.

Librarian Programs. The set of programs designed for servicing the DOS YeS libraries -- updating, copying and representing their state -- is called the librarian. The object of operation of the librarian is the types of libraries existing in the DOS YeS system: the system (the library of absolute modules, the library of target modules, the library of initial modules) and personal (the library of target modules, library of initial modules).

The system libraries enter into the resident file. The library of absolute modules is always present in it, and the other two system libraries can be absent. The personal libraries do not enter into the resident file and, as a rule, they are located on package of discs differing from the system residents. The basic admissible combinations of DOS YeS libraries are presented in Fig 22.

For each of the libraries there is a long index in which there are entries for all elements of the library with defined information about the elements of this library. The index entries are basically used to determine the location of specific elements in the library on the disc.

The sizes of the DOS YeS libraries are determined by the user during the creation of the libraries, and they depend on the number and size of the elements which must be placed in the library.

Libraries. A phase is an element of a library of absolute modules. All of the programs in the library of absolute modules have absolute format, that is, they are edited for a fixed place in the basic memory.

The library of absolute modules contains system control and service programs, translators, sorting programs and other components of the DOS YeS system delivered by the user. In addition, the library of absolute modules contains the user programs ready for execution.

The index which contains an entry with defined information about the phase for each phase corresponds to the library of absolute modules. The entries in the index were used to determine the location of the phases in the library and call them into the basic memory. The phases in the library and the entries corresponding to them in the index are arranged in the sequence in which the phases are placed by the Editor in the library.

The target module is an element of the target module library. Each module is the result of a translation by any of the DOS YeS translators. The modules can be placed in the library only by the Librarian program. The library of target modules permits the user to store frequently used modules in the system and combine them with different, newly created modules in the editing step to obtain a program ready for execution.

The library of target modules corresponds to the index for which there is an entry for each module with information about the module. The index entries are used to determine the location of the module in the library and obtain it from the library. The modules in the library and the entries corresponding to them in the index are arranged in the same sequence in which the modules appear in the library.

The element of the library of the initial modules is the initial module -- a book. A book is a series of operators in any of the DOS YeS programming languages. The basic purpose of the library of initial modules is accumulation of macrodefinitions and initial programs written in the DOS YeS programming languages. A book is the least unit with which the Librarian can operate when performing the majority of functions for the library of initial modules. The unit of the library of initial modules larger than a book is a sublibrary. This is a set of books in the source language designed for use by a defined translator. Each book in the library belongs to some sublibrary.

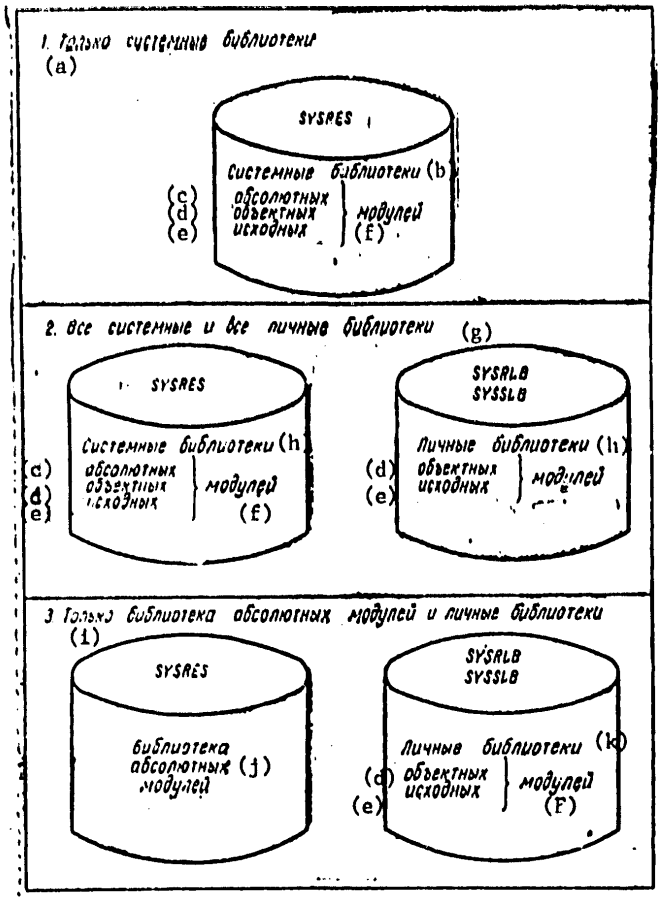

Figure 22. Basic admissible combinations of DOS YeS Libraries

Key:

a. 1. System libraries only
b. System libraries
c. Absolute
d. Target
e. Initial
f. Modules
g. 2. All system and all personal libraries
h. Personal libraries
i. 3. Only the library of absolute modules and personal libraries
j. Library of absolute modules
k. Personal libraries

The system has sublibraries for Assembler and Cobol, and the user can define and use other sublibraries. The classification of books by sublibraries permits the books written in different languages to have coinciding means.

The library index which contains for each book an entry with information about the books corresponds to the library of initial modules. The index entries are used to determine the location of the books in the library and obtain them from the library. The books in the library and the entries in the index are arranged in the same sequence in which the books appear in the library.

The possibility of creating personal libraries makes it possible for the user to have at his disposal a library of target modules and a library of initial modules in a package which is separate from the system residents. The existence of personal libraries offers a number of advantages. For example, the user has the possibility of significantly increasing the size of the library of absolute modules as a result of decreasing the volume or completely cancelling the system libraries of target and initial modules in the resident package of discs. The contents of these libraries can be partially or completely moved to the personal libraries. This is especially convenient, for the user can have several personal libraries of target or initial modules, each of which serves a specific purpose. In each specific case the programmer can indicate which of the personal libraries must be used in his assignment.

The personal libraries are also convenient for debugging inasmuch as the user has the possibility of storing altered versions of his programs in the personal libraries.

The personal libraries have internal structure analogous to the structure of the corresponding system library, but the external ones are formed of independent files.

In the DOS YeS system it is possible to use the personal libraries of the target and the initial modules in combination with the system libraries of target and initial modules.

Function of the Librarian. The functions performed by the Librarian programs can be divided into three basic classes: correction, printing and punching, creation and copying.

The functions of each of the indicated classes are realized by the corresponding programs of the librarian and extending, as a rule, to all types of libraries existing in the DOS YeS system. The indication of which of the functions and on which of the libraries it is necessary to execute, is provided by the programmer using the Librarain control operators. The correction or updating functions include the following: cataloging, removal, compression, renaming, renewal, distribution.

The cataloging procedure consists in the fact that the following are added: the phase -- in the absolute module library, the module in the target module library and the book in the initial module library. The cataloging of phases can be carried out only directly after their editing if the cataloging load is indicated for the Editor. This procedure for placing phases in the library of absolute modules is the only one in the DOS YeS system.

The removal procedure consists in the fact that the entry corresponding to the removed object (phase, module, book) is struck out of the index of the corresponding library. After removal, the object ceases to exist for the system although physically the place which it occupies in library is not released and cannot be used for other objects. This space will be considered free and available for use only after the compression procedure has been performed by the given library, the basic purpose of which is to free the spaces formed as a result of the execution of the removal procedures.

Renaming is used in order to assign new names to the objects of the required library.

The renewal operating procedure permits the contents of the individual books to be changed, and it is effective only for the libraries of initial modules.

Distribution is used in order to change the sizes of the regions on the discs allotted to the libraries and their indexes. During distribution it is possible to enlarge or diminish the libraries and their indexes and also to exclude or add an entire system library of target or initial modules. Each redistributed library is automatically compressed.

By using the print and punch functions of the Librarian, the user can output the following:

The content of specific phases, modules and books and also an entire library on the system printer, on the system punch or on both of these devices simultaneously;

The contents of the indexes of specific system or personal libraries on the system printer;

Information about the distribution of the disc packages between the libraries and their indexes on the system printer.

The punch cards obtained when punching the phases or target modules can be used as the input information for the Editor.

When using the functions for creation and copying, the user can obtain copies of the resident file or create personal libraries. The copy of the system can be complete or partial. It is possible, for example, for the partial copy to include the library of initial modules available in the original.

The user must remember that the personal libraries are serviced only in the case where the physical devices designed for the SYSSLB and SYSRLB system logical units.

The correction function and also the printing function for personal libraries are analogous to the same functions for the system libraries with the exception of the distribution procedure which is ineffective for personal libraries. The size of the personal libraries is determined during their creation and cannot be altered during the entire operating time with these personal libraries.

Debug Program. Debug is the debugging program, the basic purpose of which consists in assisting the programmer in debugging his program.

Debug offers the following basic capabilities:

Correction of the programs which will permit replacement, addition or removal of program instructions without retranslation in the target module;

Printing information of interest to the programmer during execution of the program at given points on the system printer: the contents of the common registers, the contents of the indicated regions of the basic memory;

Printing of information of interest to the programmer on satisfaction of the indicated conditions;

Printing of the contents of the basic memory on the system printer in the case of normal or abnormal completion of the assignment;

Obtaining of a list of phases with indication of addresses from which the phases were loaded into the basic memory;

The construction of files on automatic tape which the debugged program required as input files.

Any of the indicated capabilities can be requested by the programmer by using the control operators of the Debug program. The input information for Debug is the debugged program which arrives for debugging in the form of target modules. The Debug program processes the target modules similarly to the Editor and forms program phases from them which include information corresponding to the Debug request of the programmer. The program phases prepared in this way are designed for direct execution of the given assignment and cannot be cataloged in the library of absolute modules.

The program prepared for debugging is called by Debug into the basic memory and is executed under its control. During the execution of the Debug program, debugging information is output which was requested by the programmer using the control operators.

The Debug insures the greatest conveniences when debugging the program written in Assembler. Thus, in the general case in the Debug control operators hexadecimal or decimal addresses of the Debug program are indicated. For the programs written in Assembler, these addresses can be given symbolically.

The debugging is always done in the background division and requires that the entire region of problem programs be at its disposal, that is, the size of both divisions of the front plant must be zero.

Copy Programs. Independently of the specific nature of using the data processing system there are certain frequently repeated operations in the various operations. These operations can be distinguished in the parts which depend on the configuration of the computer and the format of the individual user's data, but their essence is not altered from this. These operations include, for example, the shifting of files from cards to discs or magnetic tape, printing out the discs or tape regions. In the DOS YeS system there is a set of programs which perform such functions which are called copy programs.

The copy programs edited for execution in the background division are stored in the library of absolute modules. They can be edited from any division of basic memory, for they exist in the form of target modules in the target module library. All of the copy programs can be divided into two groups: the file-file type copy programs and special programs.

The file-file type programs move one file from one data carrier to another. Exchange of information between the following data carriers is possible (in any combination):

from { cards, disc, tape } to { cards, discs, tape, printer }

Each of the copy programs of the file-file type can execute the following functions:

Coding -- file is copied without changing the entry format, size of the block: as a rule, all of the entries of the input file are present in the created copy;

Reblocking -- the file is copied with alteration only of the block size;

Recomposition -- during copying the internal structure of each entry can be altered; permutation, removal of fields, field conversion (packing, unpacking, conversion to hexadecimal form) are possible;

Reblocking with recomposition -- during copying the size of the block in the internal structure of the entry can be altered.

Special programs -- this is a set of independent programs which permit execution of the following functions:

Preparation for operation of the package of discs -- initialization of the discs;

Clearing of one or several regions of the disc;

Designation of redundant tracks to replace defective tracks of the discs;

Comparison of two tape files to establish identicalness;

Copying of the disc file or the entire volume on the disc, tape and cards.

Restoration of the initial file or volume on the disc, tape and cards.

When using any of the copy programs the programmer can, with the help of the control operators, indicate which of the functions are needed for execution of this work. The control operators are used by the copy programs in order to adjust the program for execution of the functions requested by the programmer.

Copy Macroinstructions. The copy programs, although they can be adjusted to the input file parameters and the created output file, during the copying process can only execute an entirely defined file processing. If it is requested that the processing be carried out in the copy process which has not been provided for by the corresponding copy program, the programmer can easily put together his own copy program using the copy macroinstructions.

The DOS YeS has macroinstructions for input from discs, output to the discs, input from magnetic tape, output to magnetic tape, input from cards, output to cards, output to printer, input from punch tape, output to punch tape, input from the panel typewriter, output to the panel typewriter.

The copy macroinstructions can be used when writing the panel-file type programs, the programs for information input from external carriers to the basic memory and the programs for output of information from the basic memory to the external carriers.

When writing the program it is possible to add any subroutines written by the user which can supply controlling information, process the user marks, carry out any processing of entries, add and remove entries, and so on.

The macrodefinitions corresponding to the copy macroinstructions are stored in the Assembler sublibrary of initial operators and were included in the program generated by the Assembler.

Sorting Programs. The DOS YeS service programs include the programs which offer the possibility for the user to sort the files made up of unordered entries and also to combine several files with ordered entries into one ordered, series-organized file.

The entries can be sorted in decreasing or increasing order, where ordering is carried out for each attribute called the controlling field, and its own sequence can be given. The process of combining several files, inside each of which the entries are ordered into one file of ordered entries is called merging. For the merging operations the output sequence of entries must be the same as the input sequence. If, for example, the file to be merged with other files was ordered by some attribute in increasing order, then after merging the entries of the output file will be ordered with respect to this attribute also in increasing order.

During the operation of the sorting program, check points are created which offers the possibility of using the repeat start means existing in the DOS YeS system, interrupting the sorting process when necessary and renewal of execution of the program again from the fixed intermediate point.

In order to describe the specific operation with respect to sorting or merging which must be executed, the programmer must prepare the controlling sorting operators. With their help the programmer communicates to the sorting programs such information as the description of the files to be sorted or merged, a description of the controlling fields by which the entries must be sorted; the operating mode of the program itself.

In the sorting program provisions is made for means to include the programs written by the users. The user programs which are executed together with the sorting programs can open and close files, insert, modify or remove entries, and process input and output errors.

The DOS YeS system has the following programs: the sorting program on magnetic tapes and the sorting program on magnetic tapes and discs. Each of these programs can be executed as an independent assignment in the package processing mode. In contrast to the sorting program on magentic tapes which can be executed only in the background division, the sorting program on magnetic tapes and discs is self-moving and can be executed in any division. In addition, the sorting program on magnetic tapes and discs can be included in the problem program as a subroutine.

In order to execute the sorting programs no less than 10K bytes of basic memory are required.

Sorting Program on Magnetic Tapes. The sorting program on magnetic tapes orders the input files located on magnetic tapes, it uses only magnetic tapes for the intermediate (operating) files, and it places the sorted output file also on magnetic tapes. The basic characteristics of this program are the following:

The input and output files can be multivolume;

One ordered output file can have up to nine input files sorted on it;

When using only the merging load, up to seven input ordered files can be combined into one ordered output file;

For intermediate working files from 3 to 6 units on magnetic tape are required;

Sorting can be carried out simultaneously by no more than 12 control fields, the length of each of which must not exceed 256 bytes;

The type of data contained in the controlling field must be identical for all fields and can be one of the following: binary, alphanumeric, fixed point or floating point;

The sorting program makes it possible to deal with the input file data blocks not subject to being read from magnetic tapes;

The processed files can be made up of entries of fixed or variable length which can be grouped into blocks of fixed or variable length, respectively.

The sorting program on magnetic tapes is made up of several phases which are constantly stored in the library of absolute modules. In the specific execution, the sorting program is modified in accordance with the given control operators, insuring a high sorting speed.

Sorting Program on Magnetic Tapes and Discs. In the sorting program on magnetic tapes and discs for each of the types of files (input, output and working), independently of each other, magnetic tapes and (or) discs can be used. For one type of file either magnetic tapes or discs must be used.

Depending on the type of units which were used when sorting it is possible to isolate the following versions:

Discs, when discs are used for input, working and output files;

Tape, when magnetic tapes are used for input, working and output files;

Mixed when the discs and magnetic tapes are used for input, working and output files.

The sorting program on magnetic discs and tapes provides all of the properties which were indicated for the sorting program on magnetic tapes. In addition, the sorting program on discs and magnetic tapes offers the following capability:

Indication of the control fields of different types;

Merging of up to eight ordered files into one ordered file;

Having the files available on discs and magnetic tapes;

Replacement of the sorting and the entries in the file by sorting of the control information about the entries of the file.

When realizing the last-mentioned capability, as a result of execution of the program an output file is created which is made up of the ordered control fields of entries and the disc addresses of the corresponding entries or only, the disc addresses of the entries. This address file is used to process the initial input unsorted file in the ordered sequence.

The sorting program on magnetic tapes and discs is a set of target modules in the target module library. From these modules, sorting programs can be obtained which have different capabilities (the program executing only the sorting operations; the program executing only the merging operations; the program executing both the sorting operation and the merging operation and using only discs or only magnetic tapes or magnetic tapes and discs for the working files). The specific version of the program can be cataloged in the library of absolute modules and then used without preliminary editing.

Program for Checking Unit. The programming for checking units is designed for checking the correctness of the functioning of the input-output devices, and it is executed under the control of the DOS YeS system. The test checking of the devices produced by this program can be carried out in the multiprogram mode; here the program for checking the unit itself is always executed only in the background division. This program can be executed on the machine simultaneously with the solution of other problems, the test check of the units made by it is called a nonautonomous check.

The program for checking the units offers the following capability:

Periodic preventive checking of the operation of the unit;

On appearance of errors in the operation of the unit, location of the places of the occurrence of these errors;

Conviction of the correctness of the operation of the unit after elimination of the error or introduction of technical changes into the unit.

The program for checking the units provides the test nonautonomous check of all units of the YeS EVM computers which are included in the DOS YeS system. Each specific execution of the operator must indicate which of the units is to be checked. Only the units which have been assigned at the given time to the background division can be checked.

For various types of units there are individual test programs which are adjusted, started and executed under the control of the unit test program.

No less than 10 Kbytes of basic memory are required for execution of the unit test program.

CHAPTER 5. SYSTEM GENERATION

The generation of the DOS YeS operating system is the process of obtaining a specific resident package containing the components of the DOS YeS system in the form and size as needed for daily operation of the system. The purpose of the generation is to obtain the resident package which will correspond to the highest degree to the requirements of the specific computer center, its goals and the composition of its equipment.

In complete volume the DOS YeS system is a broad set of program components, each of which is an independent module of the system. The DOS YeS system itself is constructed from such modules connected to each other by a standard procedure.

If we consider the DOS YeS system as a whole, then from the point of view of the specific computer center it is redundant: the same goal can be achieved frequently by two, three and sometimes even a larger number of different methods made available by the system. For example, it is possible to designate the input-output units available to the computer as logical units in different ways: one of the procedures is to indicate the correspondence between the logical and physical units during system generation, another method is to indicate this correspondence in the initial flow of assignments. Although there is a difference between the designation made during generation and during the input flow of assignments which consists in the duration of effect of the established correspondence, nevertheless one goal is achieved -- the correspondence between the physical and logical unit is established.

For any specific use the residence package must contain not the entire system as a whole, but a fully defined part of it. Thus, in the system there is a set of translators for many programming languages (two Assemblers, Cobol, PL/1, Basic Fortran, Fortran IV, RPG), not all of which can be used at the specific computer center. For example, Fortran IV and Cobol can be used, and in this case having the remaining translators in the resident package in daily operation means to lose a large section of memory on the disc occupied by the unused translators.

The composition of the equipment at the specific computer center can influence the composition of the programs contained on the resident package such as the copy program, the modules of the logical input-output control system. For example, if there are no punch tape units in the computer center, then the corresponding copy programs and input-output modules are not needed in the resident package. The composition of the equipment has a significant effect also on the properties of the Supervisor: the composition of the equipment determines the number of serviced input-output units and the procedure for working with the units. The Supervisor can by user request service or not service the timer, the memory protection, the decimal and floating-point arithmetic.

The organization of operations on the computer center can also influence the properties of the resident package; if it is proposed that all of the problems be solved in a single-program mode, then it is possible to have the Supervisor in the resident package without providing for multiprogram processing. There are other properties of the Supervisor which can be provided by user request. The majority of them were discussed previously when describing the properties of the Supervisor.

The capability of using generation to obtain a specific system from a general set of program components is in practice the only solution to the alternative where, on the one hand it is necessary to have a united operating system for all of the users, and on the other hand, this system must be satisfied by the specific requests of the specific user.

The DOS YeS operating system, the only one for all of the users of the YeS EVM computers, has been delivered to the users on magnetic tape which is called the distributive tape. Having the distributive tape and knowing the generation rules, the user copies the DOS YeS system from the distributive tape onto the disc and performs the generation procedure.

During generation, the basic problems such as planning the Supervisor -- selection of the properties of the specific Supervisor nucleus -- planning the library -- determination of the composition, size and contents of the libraries of a specific resident package -- planning the system files -- determination of the sizes and allocation of space in the external memory for these files must all be solved. An important event in the generation procedure is, in addition, planning this procedure itself in order to determine the clear sequence of actions, for the generation process essentially depends on the composition of the equipment used for generation and permits various versions of execution.

The procedure for generating the specific DOS YeS system is executed on the YeS EVM computer under the control of the DOS YeS operating system, and it is the operation which is subdivided into several (and sometimes quite many) steps. The generation steps are formed as a package of assignments, for the execution of which such system components are used as the Supervisor, Assignment Control, Initial Load, Editor, Librarian, and Assembler. This

set of program components is the starting version of the DOS YeS system which is necessary for execution of the generation procedure.

The DOS YeS system set up by the user is the set of three libraries: absolute, target and initial modules.

The library of absolute modules contains the Supervisor, Assignment Control, the Initial Load program, Editor, Librarian, Assembler and certain copy programs. The setup Supervisor is one of the specific versions which can be used in one of the YeS EVM computers, independently of the configuration of the specific machine. This Supervisor does not take into account the peculiarities of the specific machine or the wishes of the user. There are many properties absent in it which are desirable, and frequently necessary for many programmers. The basic purpose of this Supervisor is to provide for the execution of the system generation procedure. All of the remaining programs contained in the library of absolute modules have been edited for execution in the background division under the control of this specific Supervisor.

The library of target modules contains the target modules of all of the program components of the DOS YeS system (the Assignment Control program, Editor, Librarian, Debug, copy programs, sorting programs, translators, specific modules of the logical input-output control system). The library of initial modules contains macrodefinitions for the system macroinstructions of Assembler. Among them are the following macrodefinitions: communications with the Supervisor, generation of the Supervisor, logical input-output control system, and the copy. Here some of the check problems are contained for different service programs and translators.

The system obtained by the user can be used on his computer. However, the majority of users execute the generation procedure, as a result of which the specific Supervisor is created, taking into account the configuration of the computer and the properties requested by the user and also the system library in which only the components of the DOS YeS system needed by the user are placed.

The essence of the generation process consists in the following:

Using the set of generation macroinstructions of the Supervisor the programmer describes the Supervisor properties that he needs;

The prepared macroinstructions are translated by the Assembler, as a result of which a specific requested version of the Supervisor is obtained;

The obtained Supervisor is edited and cataloged in the library of absolute modules;

New system libraries are created.

In the generated program the nucleus of the Supervisor can be distinguished with respect to size from the nucleus of the Supervisor set up by the user, that is, can occupy more or less space in the basic memory. Therefore, as a rule, one of the generation steps is editing the system service programs and the required translators for which the new beginning of the region of the problem programs in the basic memory is taken into account. The edited system programs are cataloged in the library of absolute modules, forming its basic content. The editing process is also executed for the programs which in the system set up by the user are not in the library of absolute modules, but which are contained in the library of target modules (for example, the sorting programs, translators from Fortran, RPG).

The user can alter the contents and the sizes of the system libraries of target and initial modules in accordance with the problems which will be solved on his computer. Thus, the target modules of certain copy programs can be eliminated given the case where the use of the corresponding peripheral devices is not anticipated on the computer. The target modules of the program components which are edited and placed in the library of absolute modules are removed from the system library of target modules. The macrodefinitions of the Supervisor generation, the macrodefinitions of the logical input-output control system for the devices which are not used on the computer can be removed from the system library of initial modules. These procedures offer the possibility of economically distributing the package of discs among the system libraries. As a result of execution of the above-described procedures, the DOS YeS operating system is created for the specific computer.

As has been stated, on generation of the specific DOS YeS system, the user can determine which of the capabilities presented by the system must be insured in the system. The user can, with the help of the Supervisor generation macroinstruction, for example, indicate the following:

The type of operation of the desired DOS YeS system: the signal-program or multiprogram;

The configuration of the input-output units of his computer (number of devices and their types);

The specific features of the equipment which will be provided in the DOS YeS system: the model number of the YeS EVM computer, servicing of the timer, and organization of memory protection;

The number of logical units of the programmer for each of the divisions;

Standard designations for the logical units of the background division;

Standard operating divisions of the system (number of rows on a page and the system printer, unloading memory on abnormal completion of the assignment, keeping of statistics on magnetic tape failures, the capability of communication of the operator with the proper programs, and so on).

In executing the generation procedure, the user obtains a specific version of the resident package of discs called the operating package. This package is used in daily operation on the YeS EVM computer. The operating package is created considering the requirements not of any one specific problem, but considering the requirements of the class of problems, it is sufficiently universal, suitable for various purposes. Even in cases where certain standard properties of the operating package included during generation do not completely correspond to the requirements of the specific operation, the system has means of operative cancellation of the unsuitable standard property in the operating process. In such situations it is not necessary to alter the operating package, since the corresponding alteration is achieved on the level of the controlling operators describing the assignment. This principle is entirely justified, for the system generation procedures requires quite a lot of time, and therefore it is inexpedient to create the operating package for the specific problem. Several operating packages can be used at the computer center which are distinguished from each other by the Supervisor properties and also the composition, size and contents of the libraries. Each of these packages can be used when solving a defined class of problems.

CHAPTER 6. DEVELOPMENT OF THE DOS YeS SYSTEM

The properties of the DOS YeS operating system described in the preceding chapters cannot be considered as once and for all fixed and invariant. The continuous development of the technical means of the integrated computer system [YeS EVM], the expansion of the regions of application of the computers, the theory and practice of programming naturally require that the operating systems be developed and improved.

It is obvious that the DOS YeS operating system will only in this case satisfy the growing requirements of the users if it is constantly developed, expanding the volume of operating services. In this case one of the basic requirements that is advanced is the requirement of insuring priority among the various editions of the DOS YeS system. This means that from the point of view of the user, each next version of the system makes new capabilities available, retaining the capabilities in the preceding version. The user programs executed under the control of the preceding version of the DOS YeS system, must be executed also under the control of the next edition. Each new version of the system must be considered as a new edition or, what amounts to the same thing, the next version of the system.

The development of the DOS YeS system will proceed with respect to the following basic areas: expansion of the composition of the components, improvement of the existing components for more efficient utilization of resources and improvement of the operating reliability of the system, insurance of new technical means and simplification of the programming.

The most significant expansions which are realized at the present time in the DOS YeS system will be considered below.

Expansion of the Class of System Programs Executed in the First Plan Divisions. In order to facilitate the planning of the multiprogram processing and simplify the procedures with respect to its realization, the basic system service programs (Editor, Librarian, translators and so on) can be executed both in the background division and in the front plan divisions. For the front plan divisions it is possible to indicate the standard designations for the logical units. At the same time it becomes possible to translate and edit the programs in any of the divisions of the basic memory. This means that the special role of the background division

for the system is in practice cancelled; the procedure of formation of the flow of assignments becomes identical both for the background division and for the front plan divisions.

Personal Libraries of Absolute Modules. In addition to the personal libraries of target and initial modules existing in the system, there is the possibility of creating personal libraries of absolute modules. The user can have his own personal libraries of absolute modules for each of the divisions of the basic memory, storage programs in them and not load the system library of absolute modules with his programs. In various personal libraries the user can store the same program edited for execution in the different divisions. In this way this program can be placed in all of the personal libraries under the same name.

Multiproblem Mode. The capabilities of the multiprogram processing have been expanded by introducing the multiproblem mode. The multiproblem mode permits organization of simultaneous execution of several parallel branches of the program in the division. If it is possible to isolate several problems (branches) in the program which can be executed in paralle, then with introduction of the multiproblem mode the efficiency of the execution of the program can be approved. At the same time multiproblem mode permits us to increase the total number of simultaneously executed independent programs. The multiproblem mode means that one of the problems executed in parallel in the division is the basic problem, it is called into the basic memory and is started traditionally. During its execution the basic problem indicates to the Supervisor the necessity for starting another problem in the division in the form of the parallel process -- the subproblem of the basic problem. In the system provision is made for means for connection, disconnection and synchronizing of the parallel execution of the problems and also protection of the computer resources used simultaneously by the executed problems.

In order to create the programs in which parallel branches are isolated, the system provides the programmer with a set of macroinstructions of multiproblem regime. The programmer uses these macroinstructions in the basic problem when organizing the branch and collection points and also when indicating the points of completion of it in the subproblem. At the branch point connection of one or several parallel subproblems takes place. At the collection point the basic problem should organize waiting for completion of one or several previously connected parallel subproblems.

The system resources allocated to the division can be distributed on the subproblem and protected from their improper use in the subproblems. The distribution of resources among the subproblems and their protection is organized by the programmer using special systems means. The most tedious problem with respect to organizing the protection of the disc tracks from the simultaneous use of them by several programs executed in the multiprogram regime is realized by the system itself. These systems means are included in the Supervisor and the logical input-output control system.

New Capabilities of Error Processing. In order to improve the system reliability and improve its operating characteristics, additional capabilities have been introduced for differentiation of machine errors, isolation of those of them which can be processed without conversion of the computer into the serious halt state. This permits continuation of the execution of the assignments not affected by the machine error.

Keeping of Statistics. The operating system of the DOS YeS system includes the special means of gathering, storing and processing statistical information which give an estimate of the fitness of the computer system and executes certain other functions. The system realizes the accumulation of statistical information about the equipment and program errors, it keeps account of the assignments and also has means for output of the accumulated statistical information in a form convenient for its use.

The gathering of the statistical information is realized by the controlling program of the DOS YeS system on request by the user; the collected information is fixed in a special register file of the system which is located on the disc. Here any event is fixed in the register file with indication of the situation in which it occurred: the division in which the program is executed, the name of the program, the type of error, the number of the device, the bytes of the basic and the more precisely defined state, and so on. It must be emphasized that the system not only records the detected erroneous situation but also independently takes measures to eliminate the consequences of the error and restore the fitness of the system.

The processing of the information accumulated in the register file system is realized using special service programs which permit us to obtain summary information about the functioning of the specific unit or program. When analyzing the data obtained, it is possible to draw the conclusion of the loading and the rhythm of the use of the equipment, the frequency of the occurrence of failures, the quality of volumes on the magnetic tapes and discs, and so on, and on the basis of this, to generate specific recommendations with respect to operation and modification of the hardware and software of the computer.

Remote Processing Means. One of the prospective areas of use of the YeS EVM computer is creation of the remote processing systems. In order to support the work with the telecommunication units of the YeS EVM computer, the DOS YeS system includes the Basic Telecommunication Method of Access which when used makes it possible to construct the specific remote processing systems of the type of data collection, the commutation of messages and the processing of requests.

The Basic Telecommunications Method of Access is a program means constructed by the general principles of the input-output control system. It realizes the standard procedures for receiving and transmitting messages over the communication lines from the remote processing units to the basic memory

and back. Here the method takes on itself the establishment of communications with the subscriber, the execution of the channel programs [drivers], transmitting the messages, the control of the buffers in the basic memory, the recording of messages from one code to another, the processing and reporting of the errors, and testing of the telecommunications units.

Support of the YeS-5061 Unit. The constant expansion and improvement of the technical means of the YeS EVM computers require corresponding modification of the DOS YeS operating system. The most significant developments in the system have been made in order to provide magnetic disc storage elements with a capacity of 29 million bytes (the YeS-5061 unit). The DOS YeS system makes available the capability of using this device as an ordinary input-output unit and also as a system unit including the use of it as system residents.

Other Expansions of the DOS YeS System. In connection with the expansion of nomenclature of external devices of the YeS EVM computers, the corresponding modification and expansion of the input-output control system of the DOS YeS system has been carried out and new capabilities have been included in it. For example, the system provides repeated entry in the input-output modules, which permits use of the same module by several subproblems when working in the multiproblem mode; the direct access method will provide for the processing of variable-length entries and access to the entry by its relative address.

The DOS YeS system Librarian performs the functions of merging the libraries which will permit exchange of elements (phases, modules and books) among like personal and system libraries.

Some changes have been made in the DOS YeS system languages and translators. These changes are primarily directed at improving the quality of the created programs and reduction of certain restrictions.

The authors have not stated the goal of providing specific information about all of the changes and expansions of the DOS YeS system. Only the most significant properties which have been realized in recent times are mentioned in this chapter.

10845
CSO: 8344/1089

- END -